

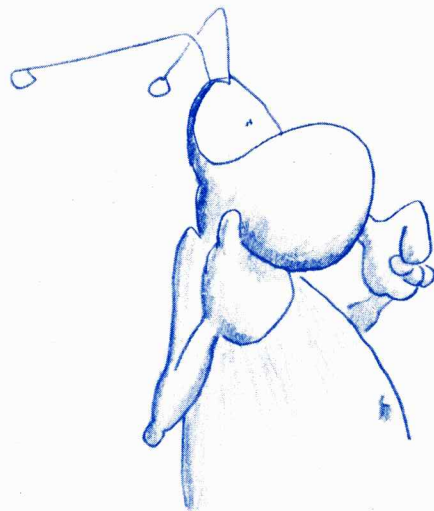
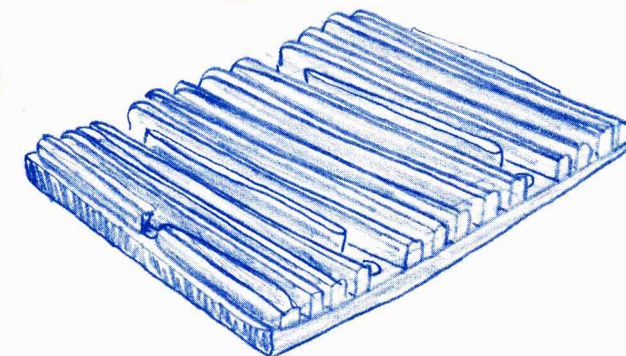
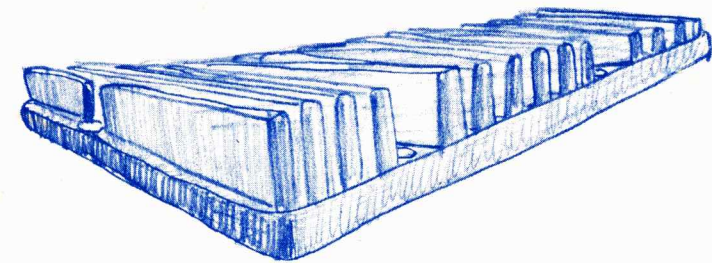
THE

HOW THEY

DO DAT

MANUAL

3pc, 1oc, EMC

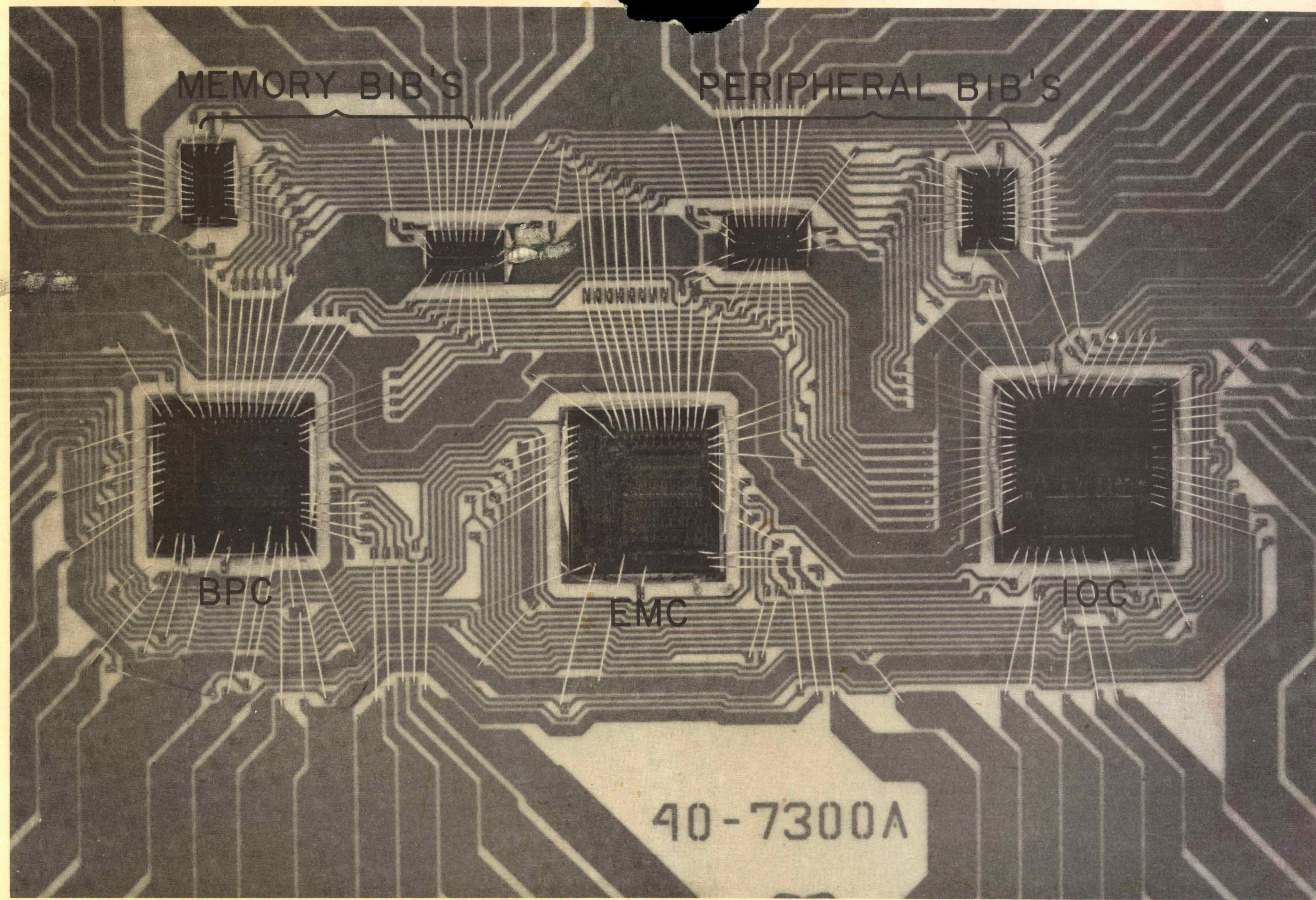


Issued to: Fred Weninger

DATE 4/4/78



54



MEMORY BIB'S

PERIPHERAL BIB'S

BPC

EMC

IOG

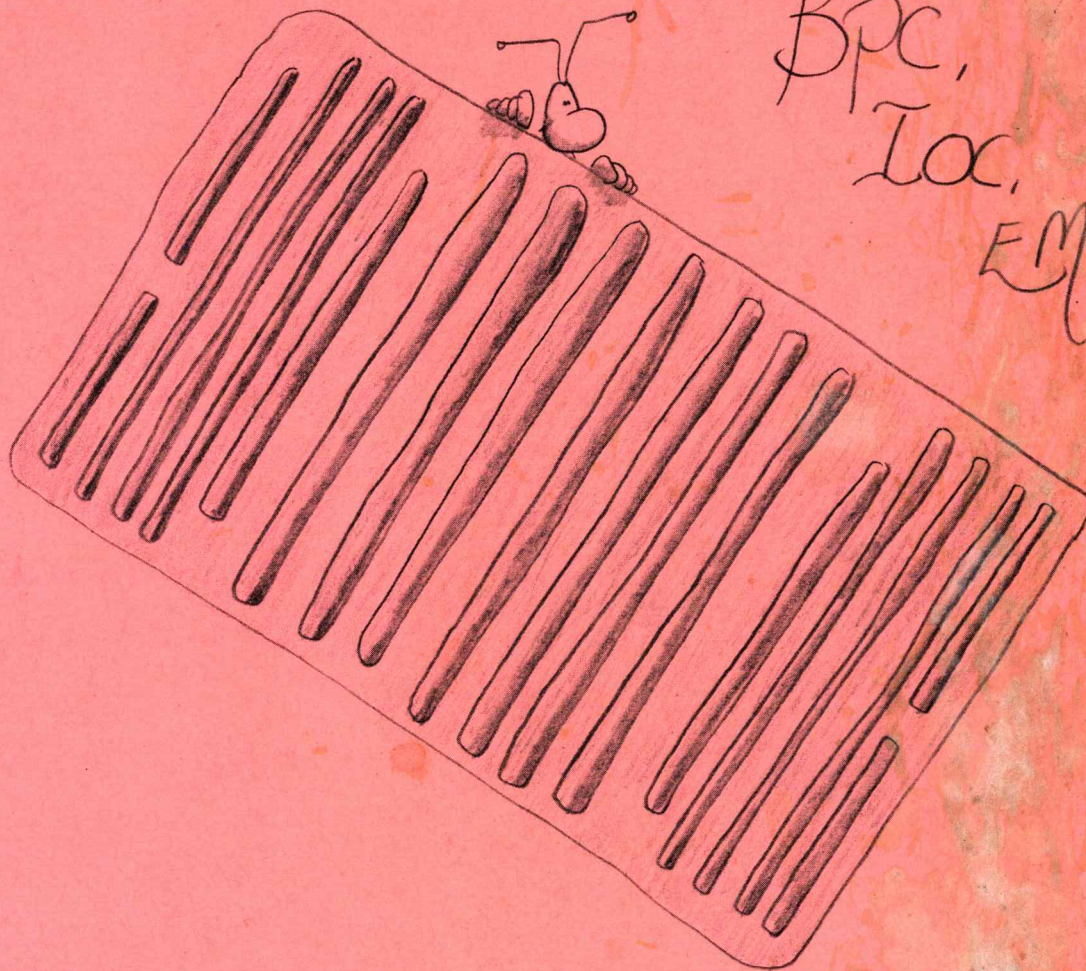
40-7300A

INTRODUCTION:

BPC,

Loc,

EMC.



4/5/78

What's in this book are the equivalent schematics for the three processor chips known as the BPC, IOC and EMC. Collectively, these chips are known as the hybrid micro-processor, also sometimes known as the CPD Processor. All are LSI chips fabricated with the N-MOS II process. The book is divided into four sections separated by colored dividers. The first section is this introduction; the remaining sections deal individually with the aforementioned chips. In the near future we plan to add another section dealing with the Address Extension Chip (AEC).

The primary purpose of this book is to give aid and comfort to those designers who have to build things using the processor. And while this book is not really an interface book (we hope to do *that* as soon as the AEC is done), it does give the best run-down possible on the in's and out's of the system definition, particularly in those areas concerning M-Section operation.

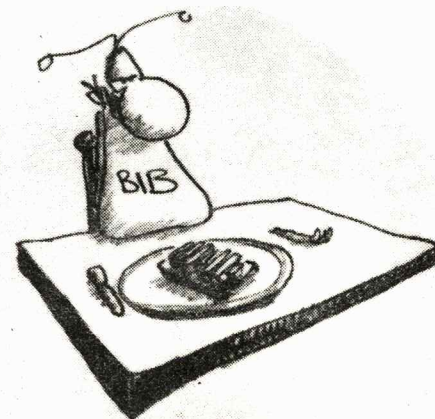
We have done the best we can to make it easy to find things in the book. Each section dealing with a chip has an index. This index takes the form of page numbers associated with the various portions of the block diagram for that chip. Questions concerning the operation of a given portion of the chip can be answered by turning to the page number indicated for that area by the block diagram. These indexing block diagrams are located a page or so behind the colored divider at the start of the section. In addition, the right-hand edge of each right-hand page contains an abbreviated listing of the topics covered for the associated pair of pages. Hopefully, this will make thumbing through the book a little easier. Humble apologies are offered for the horrible numbering scheme used for the figures. Each major topic gets the next consecutive number, which is used as the left-hand portion of the figure number. Each topic is subdivided by dashed numbers to the right of the left-hand number. Frequently, subtopics are further subdivided and sometimes even the sub-subtopic are subdivided. We got started with this scheme along time ago and it was simply too much trouble

to change it. Drawings that have or a -F as part of their drawing number indicate that they pertain on to the 15- or 16-bit version, respectively. Likewise, the abbreviated topics at the right-hand edge sometimes contain an S or F within a stretched diamond. This is a way of indicating the same thing. A drawing or section number that does not contain a -S or -F pertains to both versions. Beginning with the section on the BPC, page numbers in the book run consecutively; drawing numbers, however, start over for each section.

Within each section the topics are covered more or less in a functional sequence, with an instruction fetch serving as the model. First the connection between the external IDA Bus and the internal IDA Bus is covered. This is followed by a look at the Instruction Register and Instruction Decode. Generally, the ROM's are next, followed by any special purpose non M-Section hardware that the chip contains. This is followed by discussions of the M-Section, the ASM charts and the waveforms. The last section of each chip is a listing of its ROM.

The waveforms are quite idealized, and while they provide detailed information about the logical sequence of events and the logical options possible during operation they do not contain any data concerning actual delays or rise times. Such information will be forthcoming in the long awaited interfacing manual.

The schematic segments shown in this book are not of the same sort



THE BIB

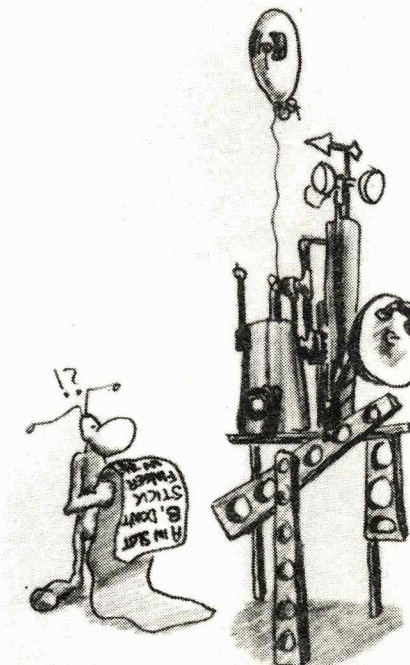
schematics used by the designers. Schematics have been redrawn and put together in a way to emphasize the functions of the various elements within the chip. Such things as location in the layout, information about device geometry, and whether or not an inverter is a standard inverter or a quasi push-pull inverter have not been included. Another thing that should be borne in mind is that all AND gates and OR gates that are shown in the schematics are really constructed from NOR gates. AND's and OR's were used to enhance the logical clarity of the schematics. Very often, for instance, the inverter shown in a signal going to an AND gate is really part of the NOR gate used in the physical implementation.

Complaints, corrections and just plain general comments concerning this book can be directed to Ed Miller of Loveland CPD. Actually, this division is about to move to Fort Collins, but I don't know what they're going to call us after we get there.

Copies of this book may be obtained by contacting Ed Miller (the author) or Judy Grimes (the illustrator who made most of the drawings over the last two years,

and who put the book together).

Also I would like to express gratitude to our cartoonist, Rand Renfro, who risked his reputation by carrying out some of the suggestions pressed upon him. He told me he was afraid that neither one of us might have a job after this thing came out.



ASSEMBLER

PROCESSOR BUG LIST

CHIP	DESCRIPTION	VERSION	
		15-bit	16-bit
BPC	- EXE of, or instruction fetch from, an addressable register in the BPC fails to give SMC	✓	
IOC	- DDR not reliable	✓	
	- IOC releases INT at wrong time to allow single level indirect for interrupt vector	✓	
	- IOC doesn't allow IOC machine-instructions to be fetched from its own registers	✓	✓
	- Glitch on BYTE	✓	✓
	- Pulse Count Mode unuseable due to "timing difficulties"	✓	---
EMC	- Multiplication with -32768 is not commutative	✓	✓
	- CMX not useable with DMA	✓	
ALL	- POP synchronizer is unreliable?	---	✓

INTRO

4-40 LOCK NUTS

HEAT SINK
(0050-0631)

HYBRID SUBSTRATE

SUBSTRATE LID

INSULATING GASKET WITH
ELASTOMATE CONNECTORS
(09825-67908)

P.C. BOARD

4-40 STUDS

* AT ASSEMBLY HYBRID AND HEAT SINK ARE BONDED TOGETHER. PART NUMBERS FOR REPLACEMENT HYBRIDS INCLUDE REPLACEMENT HEAT SINKS.

09825-67909 = 09825-67907 (15-BIT HYBRID + HEAT SINK) AND GASKET WITH ELASTOMATE CONNECTORS.

NOTCH KEYS SUBSTRATE TO HEAT SINK

15-BIT ADDRESSING
(09825-67907)*

16-BIT ADDRESSING
(2 CHIPS - T93708 (LESS EMC))*
(3 CHIPS - T59793 (WITH EMC))*

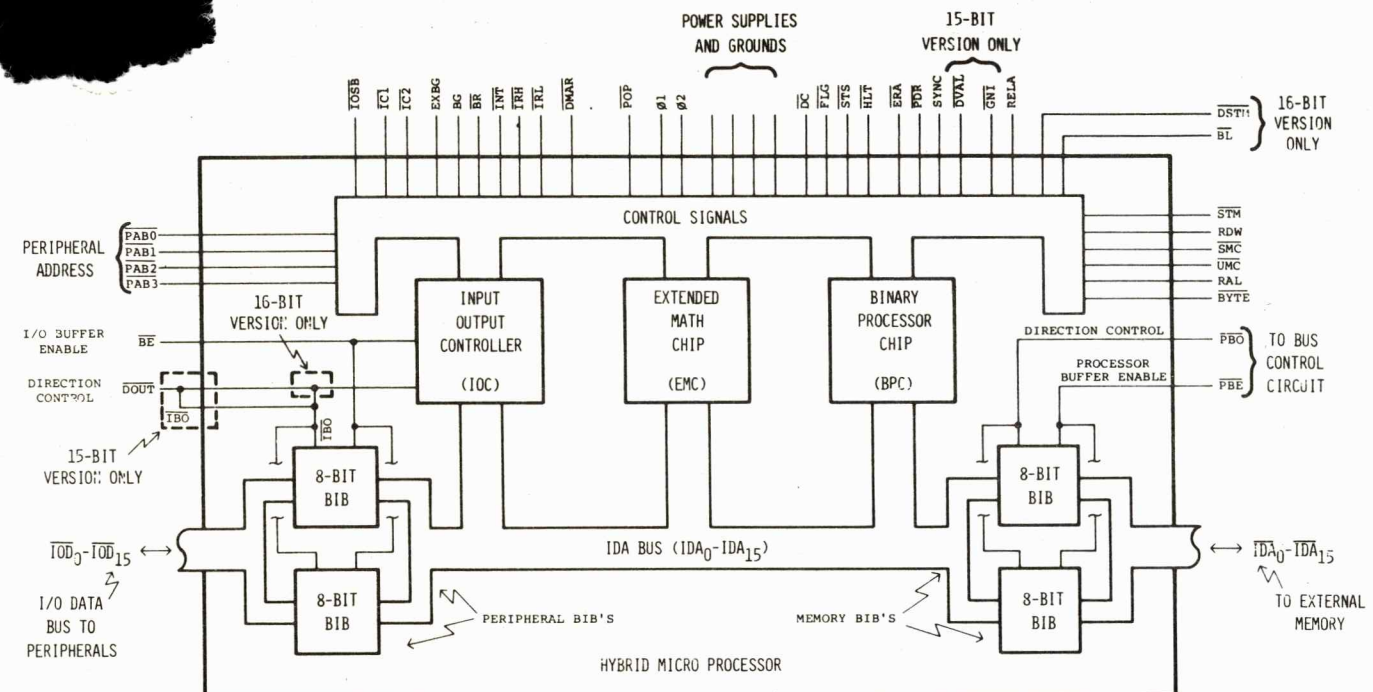
SEALED AT FACTORY

INSULATING MEDIUM

CIRCULAR 3MM GOLD-PLATED CONTACTS ON 7MM CENTERS

RUBBER CENTER FOR COMPRESSIBILITY

CLEARANCE HOLE FOR SUBSTRATE LID



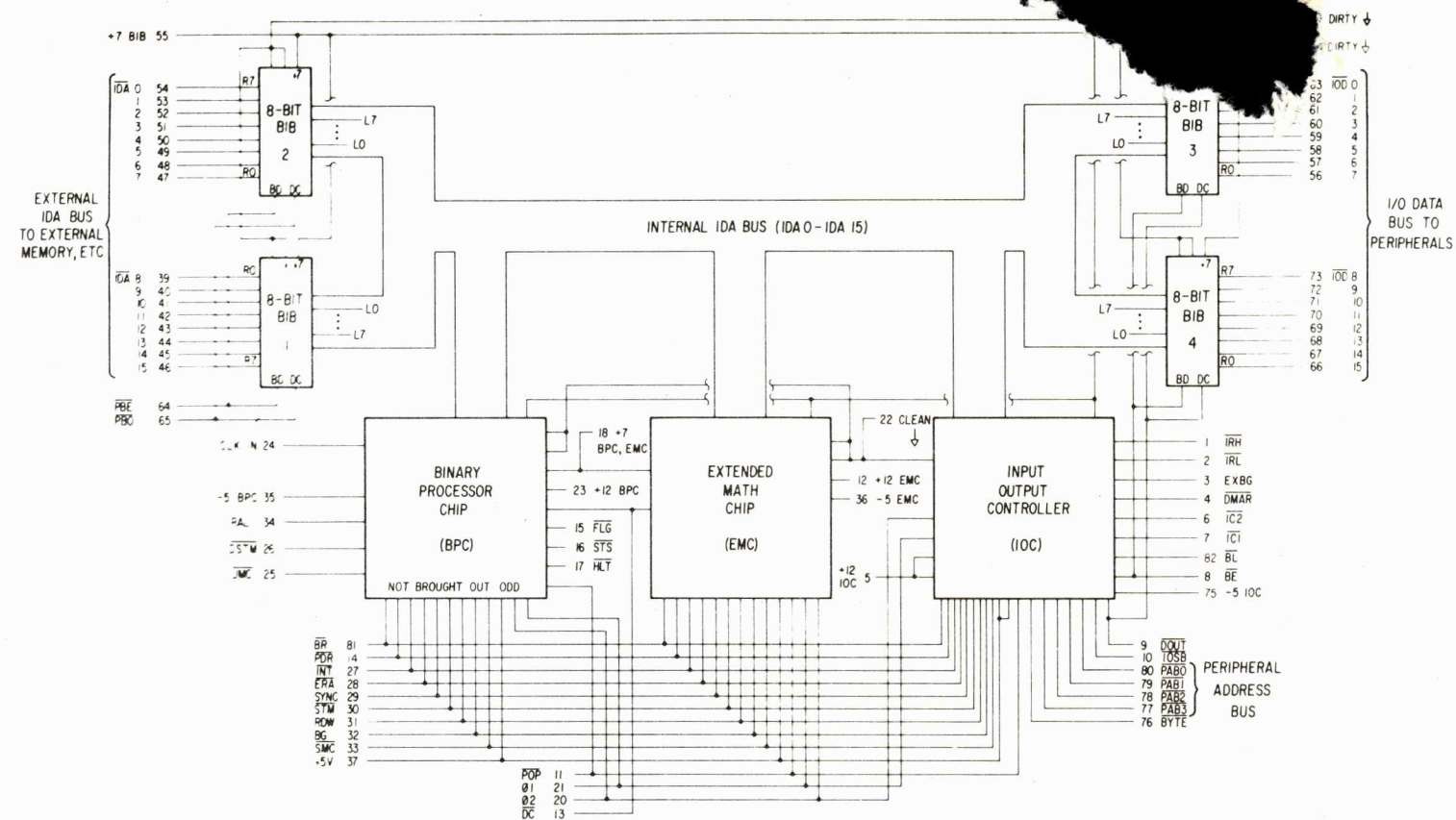
15/16 BIT PROCESSOR PIN ASSIGNMENTS

PIN #	SIGNAL	PIN #	SIGNAL	PIN #	SIGNAL	PIN #	SIGNAL
1	IRH	22	CLEAN GND	42	IDA 11	64	PBE
2	IRL	23	+12 BPC	43	IDA 12	65	PBO
3	EXBG	24	GNI ¹	44	IDA 13	66	IOD 15
4	DMAR	24	CLK IN ²	45	IDA 14	67	IOD 14
5	+12 IOC	25	UMC	46	IDA 15	68	IOD 13
6	IC 2	26	DVAL ¹	47	IDA 7	69	IOD 12
7	IC 1	26	DSTM ²	48	IDA 6	70	IOD 11
8	BE	27	INT	49	IDA 5	71	IOD 10
9	DOUT	28	ERA	50	IDA 4	72	IOD 9
10	IOSB	29	SYNC	51	IDA 3	73	IOD 8
11	POP	30	STM	52	IDA 2	74	IBO ¹
12	+12 EMC	31	RDW	53	IDA 1	74	DIRTY GND ²
13	DC	32	BG	54	IDA 0	75	-5 IOC
14	RELA ¹	33	SMC	55	+7 B1B	76	BYTE
14	PDR ²	34	RAL	56	IOD 7	77	PAB3
15	FLG	35	-5 BPC	57	IOD 6	78	PAB2
16	STS	36	-5 EMC	58	IOD 5	79	PAB1
17	HLT	37	+5	59	IOD 4	80	PAB0
18	+7 BPC, EMC	38	DIRTY GND	60	IOD 3	81	BR
19	GUARD ^{1,3}	39	IDA 8	61	IOD 2	82	PDR ¹
20	Ø2	40	IDA 9	62	IOD 1	82	BL ²
21	Ø1	41	IDA 10	63	IOD 0		

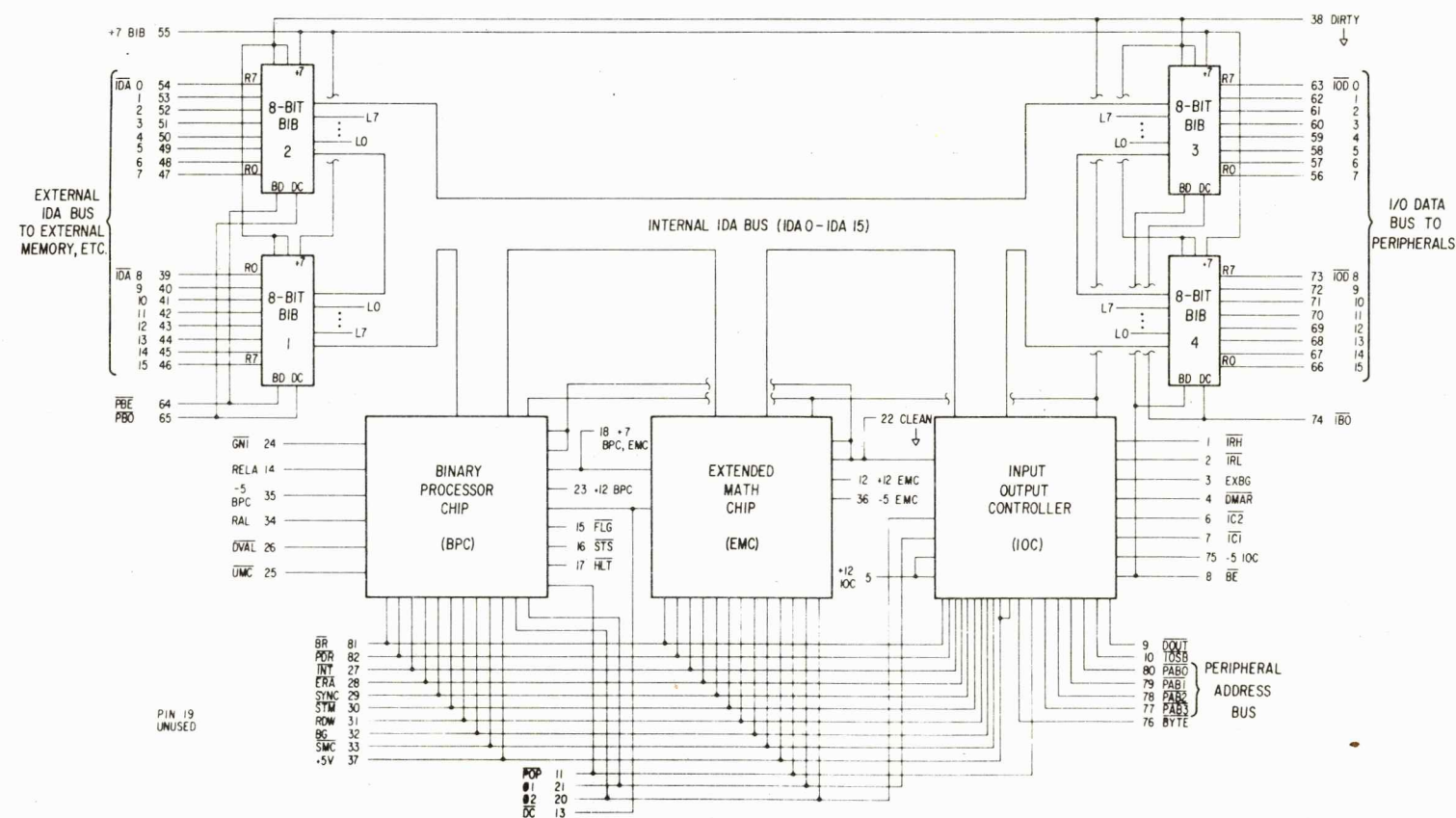
NOTE 1 15-BIT VERSION ONLY

NOTE 2 16-BIT VERSION ONLY

NOTE 3 MUST BE EXTERNALLY CONNECTED TO CLEAN GROUND

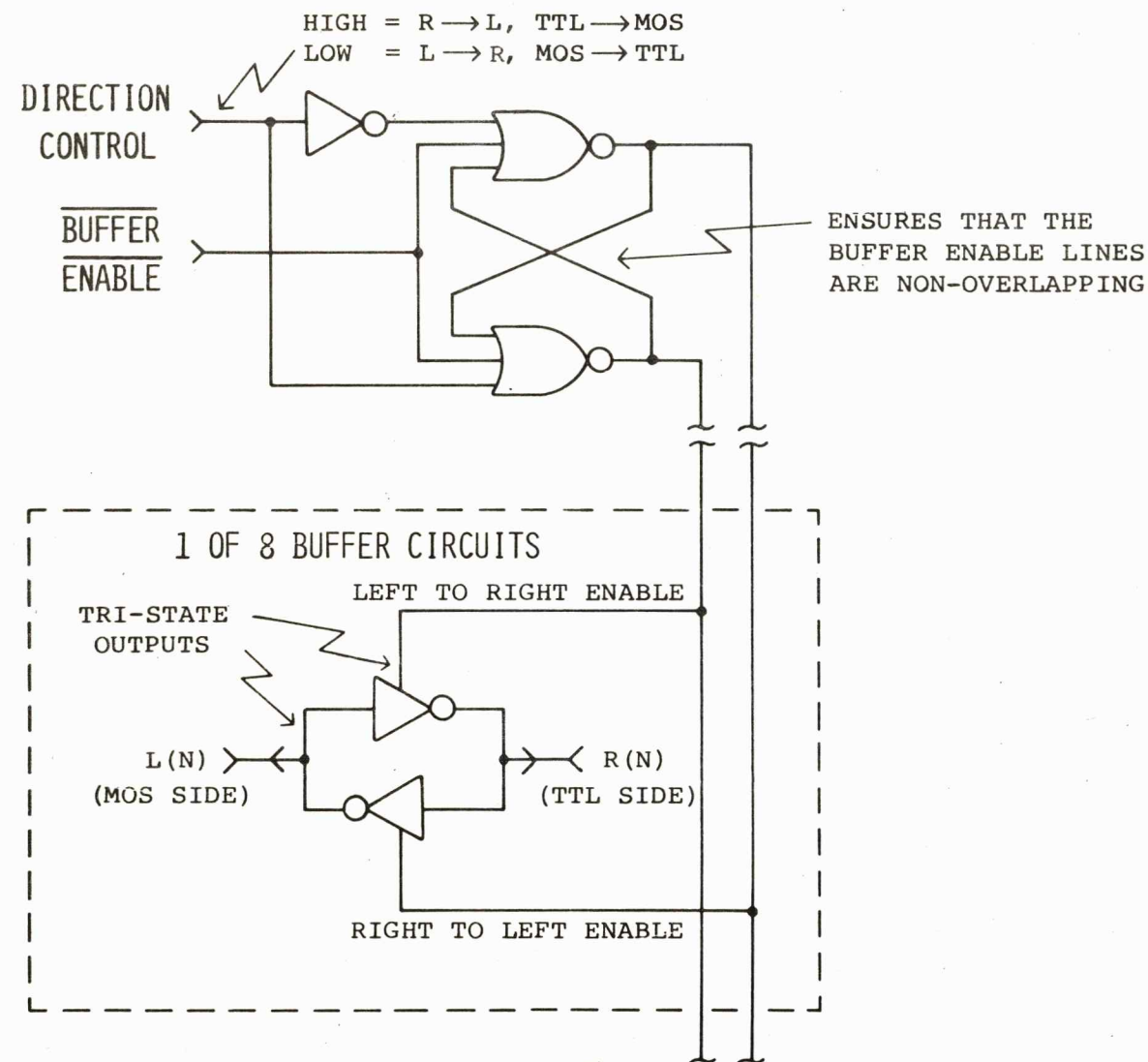


16-BIT VERSION



15-BIT VERSION

INTRO

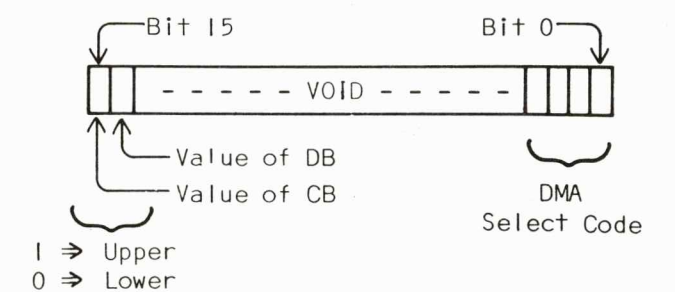


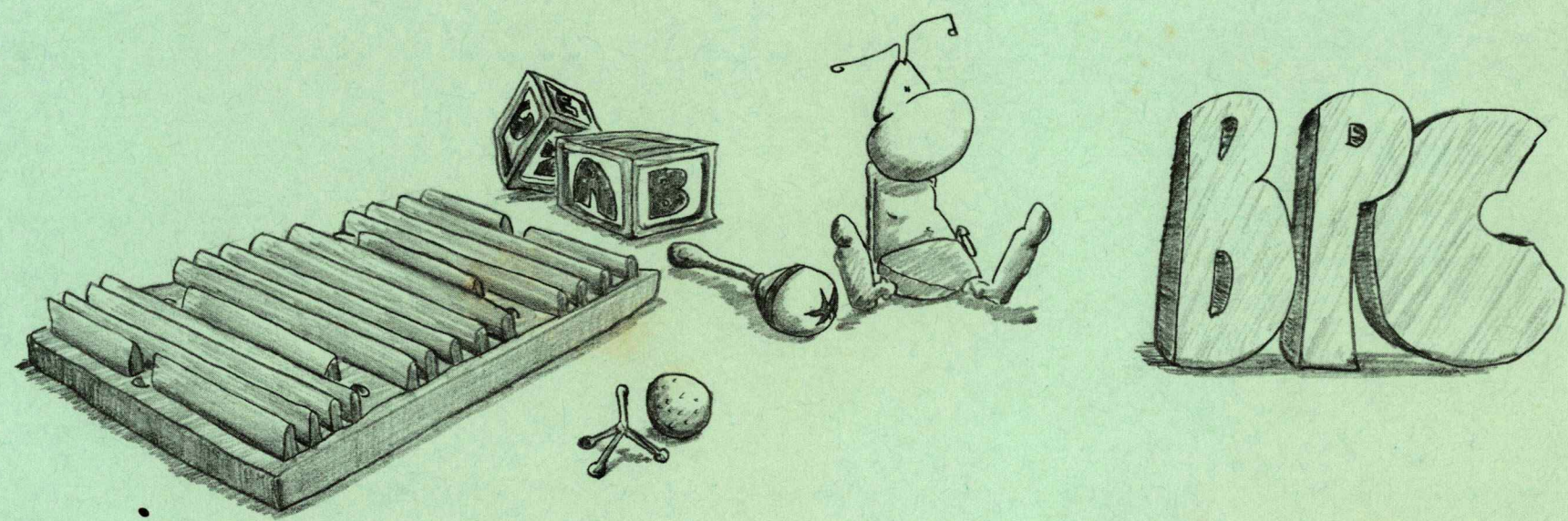
Octal Address	Name	Location	Description (# of Bits)
0	A	BPC	Arithmetic Accumulator (16)
1	B	BPC	Arithmetic Accumulator (16)
2	P	BPC	Program Location Counter (least 15 of 16 or 16)
3	R	BPC	Return Stack Pointer (least 15 of 16 or 16)
4	R4	IOC	Peripheral Activity Designator (—)
5	R5	IOC	Peripheral Activity Designation (—)
6	R6	IOC	Peripheral Activity Designator (—)
7	R7	IOC	Peripheral Activity Designator (—)
10	IV	IOC	Interrupt Vector (upper 12 of 16)
* → 11	PA	IOC	Peripheral Address Register (least 4 of 16)
12	W	IOC	Working Register (16)
† → 13	DMA PA	IOC	2 MSB = CB & DB; 4 LSB = DMA Periph. Add. Reg.
14	DMA MA	IOC	DMA Memory Address & Direction Register (16)
15	DMA C	IOC	DMA Count Register (16)
16	C	IOC	Stack Pointer (16)
7	D	IOC	Stack Pointer (16)
20-23	AR2	EMC	BCD Arithmetic Accumulator (4 x 16)
24	SE	EMC	Shift Extend Register (least 4 of 16)
* → 25-27	X	EMC	Internal Arithmetic Register (3 X 16)
30-37	UNASSIGNED		
77770/ 177770	ARI	R/W	BCD Arithmetic Register (4 x 16)

* Not available for general use. Part of processes internal to a chip.

† Read register 13₈ produces:

CB and DB are actually discrete registers, and while they can only be read by reading R13, storing into R13 will not alter their values. Use the CBL, CBU, DBL and DBU machine instructions for that purpose.





4/15/78

SECTION 1

Figure 1 is a combined block diagram of the 15 and 16-bit versions of the BPC. The majority of activity within the BPC is controlled by a ROM. This is a programmed logic array whose input qualifiers are a 4-bit state-count, group, miscellaneous, and input-output qualifiers. From the ROM are decoded micro-instructions. Each machine-instruction that the BPC executes, and the BPC's response to memory cycles directed at its addressable registers, is a complex series of micro-instructions. This activity is represented by the flow charts depicted in Section 18.

Changes in the state-count correspond to the step-by-step sequence of activity shown in the flow charts. The State-Counter has a natural sequence that was chosen by computer simulation to reduce the complexity of the necessary number of non-sequential transitions. When a section of the flow chart requires a non-sequential transition it decodes a special micro-instruction whose purpose is to override the natural sequence and produce the desired alteration in the state-count.

The group qualifiers are generated by Instruction Decode. The group qualifiers represent the class of instruction that has been fetched and that must now be executed.

The input/output qualifiers are controlled by the M-Section. Those qualifiers are used in decoding micro-instructions, and in making flow chart branches, that are dependent upon or have to do with input and output to the BPC.

The IDB Bus is the internal BPC representation of the IDA Bus. To conserve power this bus is used dynamically; it is precharged on phase two, and is available for data transmission only during phase one. (Phase one - $\phi 1$, and phase two - $\phi 2$, are two complementary non-overlapping clocks that are required by most elements in the system.) Data on the IDB Bus is transmitted in negative true form; a logical one is encoded on a given line of the bus by grounding that line.

The main means of inter-register communication within the

BPC is via the IDB Bus and the various set and dump micro-instructions. For instance, a SET I loads the I register with the contents of the IDB Bus. A DMP IDA places the contents of the IDA Bus onto the IDB Bus. A simultaneous DMP IDA and SET I loads the I register with the word encoded on the IDA Bus. As a further instance, that very activity is part of what is decoded from the ROM at the conclusion of a memory cycle that is an instruction fetch. Figure 19-10 illustrates the waveforms associated with the start-up sequence and an instruction fetch.

Once the instruction is in the I register, the bit pattern of the instruction is felt by Instruction Decode. Aside from the aforementioned group qualifiers, Instruction Decode generates two other groups of signals. One of these groups is control lines that go to the Flag Multiplexer to determine which, if any, of the external flag lines is involved in the execution of the current machine-instruction. The remaining group of signals is called the asynchronous control lines. These are signals that, unlike micro-instructions, are steady-state signals present the entire time that the machine-instruction is in the I register. The asynchronous control lines are used to determine the various modes in which much of the remaining hardware will operate during the execution of the machine-instruction. For example, the S register is capable of several types of shifting operations, and the micro-instruction that causes S to shift (SSE) means only that S should now shift one time. The exact nature of the particular type of shift to be done corresponds to the type of shift machine-instruction in the I register. This in turn affects Instruction Decode and the asynchronous control lines, which in turn affect the circuitry called S Register Shift Control. It is that circuitry that determines the particular type of shift operation that S will perform when an SSE is given.

In a similar way the asyn-

chronous control lines affect the nature of the operation of the Arithmetic-Logic Unit (ALU), the Skip Matrix, and the A and B registers.

The least four bits of the I register are a binary decremter and CTQ Qualifier network. This circuitry is used in conjunction with machine-instructions that involve shift operations. Such machine-instructions have the number of shifts to be performed encoded in their least four bits. When such an instruction is in the I register, the least four bits are decremented once for each shift that is performed.

The A and B Registers are primarily involved in machine-instructions that: read to or write from memory; do binary arithmetic; shift; or, branch. Machine-instructions that simply read from or write to memory are relatively easily executed, as the main activity consists of dumping or setting the A or B Register. The arithmetic instructions involve the ALU.

The ALU has three inputs. One is the ZAB Bus. This bus can transmit either zero, the A Register, or the B Register. The choice is determined by the asynchronous control lines. The input from the ZAB Bus can be understood in its true form or in its complemented form. The second input to the ALU is the S Register. The remaining input is a carry-in signal.

The ALU can perform three basic operations: logical AND, logical inclusive OR, and binary addition. The choice is determined by the asynchronous control lines.

Whatever operation to be performed is done between the complemented or uncomplemented contents of the ZAB Bus, and the contents of the S register. The output of the ALU is available through the DMP ALU micro-instruction, as well as through lines representing the carry-out from the 14th and 15th bits of the result. These carry-outs are used to determine whether or not to set the one-bit Extend and Overflow registers.

The R register is the return stack pointer for the RET machine-instruction.

The P register is the program counter. Associated with it are several other pieces of circuitry used for incrementing the program counter, as well as for forming complete 15 or 16-bit addresses for memory cycles needed in the execution of memory reference or skip machine-instructions. These other pieces of circuitry are the T register, the P-Adder Input, P-Adder Control, and the P-Adder.

The P-Adder mechanism can operate in one of three modes. These modes are established by micro-instructions rather than by the asynchronous control lines. In the memory reference machine-instruction mode (established for the duration of the ADM micro-instruction) the T register will contain a duplicate copy of the memory reference machine-instruction being executed. Thus, the 10-bit address field of the machine-instruction and the base page bit (bit 10), as well as top 6 bits of the program counter, are all available to the adder mechanism. In accordance with the rules of either relative or absolute addressing (as determined by RELA) the P-Adder Input and P-Adder operate to produce the correct full 15 or 16-bit address needed for the associated memory cycle.

The ADS micro-instruction establishes a mode wherein only the least five bits of a skip machine-instruction are combined with the program counter to produce a new value for the program counter.

In the absence of either an ADM or ADS micro-instruction the P-Adder mechanism defaults to an increment-P mode. In this mode the value of P+1 is continuously being formed. This is the typical way in which the value of the program counter is changed at the end of non-branching machine-instructions.

The output of the P-Adder mechanism is available to the IDB Bus through the DMP PAD micro-instruction.

SECTION 1 (CONTINUED)

The D Register is used to drive the IDA Bus through the SET IDA micro-instruction. Because of limitations on transistor device size, and the large capacitances possible on the IDA Bus, two consecutive SET IDA's are required to ensure that the IDA Bus properly represents the desired data.

The BPC has special circuitry to detect a machine-instruction that requires an indirect memory cycle. This circuitry generates a qualifier (IND) used in the ROM. Flow-charting corresponding to a machine-instruction that can do indirect addressing has special activity to handle the occurrence of an indirect reference.

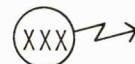
In the event of an interrupt request generated by the IOC the BPC aborts the execution of the machine-instruction just fetched, and without incrementing the program counter, executes the following machine-instruction instead: JMP 10₈, I. Register 10₈ is the Interrupt Vector register (IV) in the IOC. This is part of the means by which vectored interrupt is implemented. Figure 19-12 illustrates interrupt operation.

In the event that an addressable register within the BPC is the object of a memory cycle, whether the memory cycle is originated by the BPC itself, or by an agency external to the BPC, a BPC Register Detection and Address Latch circuit detects that fact (by the value of the address) and latches the address. It also latches whether the operation is a read or a write. The result of this action is two-fold: First, it supplies qualifier information to the ROM so that micro-instructions necessary for the completion of the memory cycle may be issued. Secondly, it initiates action within the M-Section that aids in the handling of the various memory cycle control signals.

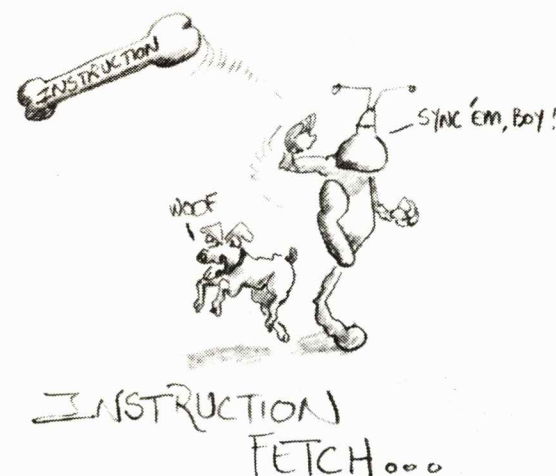
Section 19 contains a variety of waveforms that illustrate the various memory cycles that can occur in various operational contexts. Figure 19-1 explains the notational conventions used in those waveforms.

The BPC can interrupt the execution of a machine-instruction to allow some other agency to use the IDA Bus. The BPC will do this whenever Bus Request (BR) is active, and the BPC is not in the middle of a memory cycle. When these conditions are met, the BPC issues a signal called Bus Grant (BG) to inform the requesting agency that the IDA Bus is available. The BPC also generates an internal signal called Stop (STP) that halts the operation of the decremter in the I register, and halts the changing of the ROM State-Counter. In addition, STP inhibits the decoding from the ROM of all but those micro-instructions needed to respond to memory cycles, under the control of the M-Section. STP and BG are given until the requesting agency signals that its use of the IDA Bus is over by releasing BR. This capability is the basis of Direct Memory Access, as implemented by the IOC. Figure 19-13 illustrates the operation of Bus Request and Bus Grant.

Numbers in circles

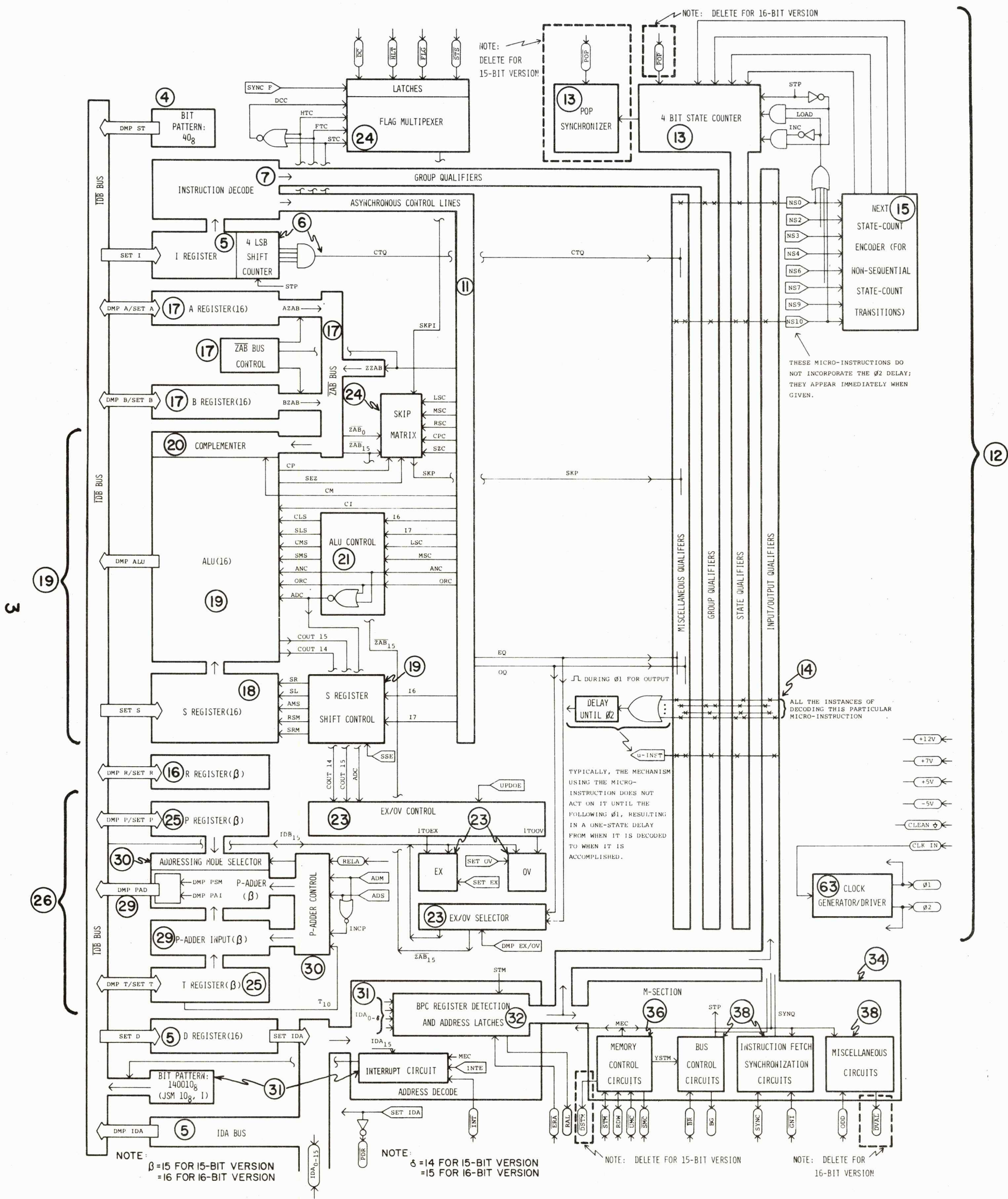


refer to the page number at which the referenced item is discussed.



NOTES FOR FIG. 1

1. DENOTES A MICRO-INSTRUCTION DECODED IN THE ROM.
2. AND DENOTE ONE- AND TWO-WAY INTERCONNECTIONS TO A BUS; ALWAYS CONTROLLED BY A ROM MICRO-INSTRUCTION.
3. DENOTES A DIRECT CONNECTION BETWEEN TWO ITEMS.
4. DENOTES A CONNECTION BETWEEN TWO ITEMS THAT IS ACTIVE ONLY WHEN THE STATED SIGNAL IS GIVEN. SUCH SIGNALS ARE NOT ROM DECODED MICRO-INSTRUCTIONS. SOME ARE PRESENT THROUGHOUT AN ENTIRE EXECUTION CYCLE, WHILE OTHERS REFLECT MORE TEMPORARY CONDITIONS.
AZAB
OR
5. DENOTES THAT THE STATED LINE REPRESENTS A DECODED CONDITION.
6. REPRESENTS A NON-MICRO-INSTRUCTION CONTROL LINE OR SOME OTHER SIGNAL.
7. REPRESENTS AN INPUT TERMINAL TO THE BPC
8. REPRESENTS AN OUTPUT TERMINAL FROM THE BPC
9. REPRESENTS A TERMINAL THAT IS BOTH AN INPUT AND AN OUTPUT.
10. NUMBERS IN PARENTHESES INDICATE THE NUMBER OF BITS A MECHANISM HANDLES.
11. THE LOGICAL SENSE (XXX VERSUS XXX) OF THE I/O TERMINALS IS CORRECTLY INDICATED. HOWEVER, THE DRAWING IS NOT A RELIABLE INDICATOR OF THE EXACT SENSE OF THE INTERNAL SIGNALS. TYPICALLY BOTH SENSES EXIST, AND FREQUENTLY THE PHYSICAL PROXIMITY OF SIGNALS TO THEIR DESTINATIONS WAS MORE IMPORTANT IN DECIDING WHICH SENSE TO USE, RATHER THAN AGREEMENT OF LOGICAL SENSE.
BECAUSE STRICT ACCURACY IN REPORTING SIGNAL SENSES ON SUCH A GENERAL LEVEL DRAWING WOULD SHARPLY INCREASE THE NUMBER OF INTERCONNECTIONS, WITH ONLY A SLIGHT INCREASE IN USEFULNESS, WE USUALLY SHOW ONLY THE NAME OF THE SIGNAL.



OVERVIEW OF THE IDA BUS - IDB BUS INTERCONNECTION

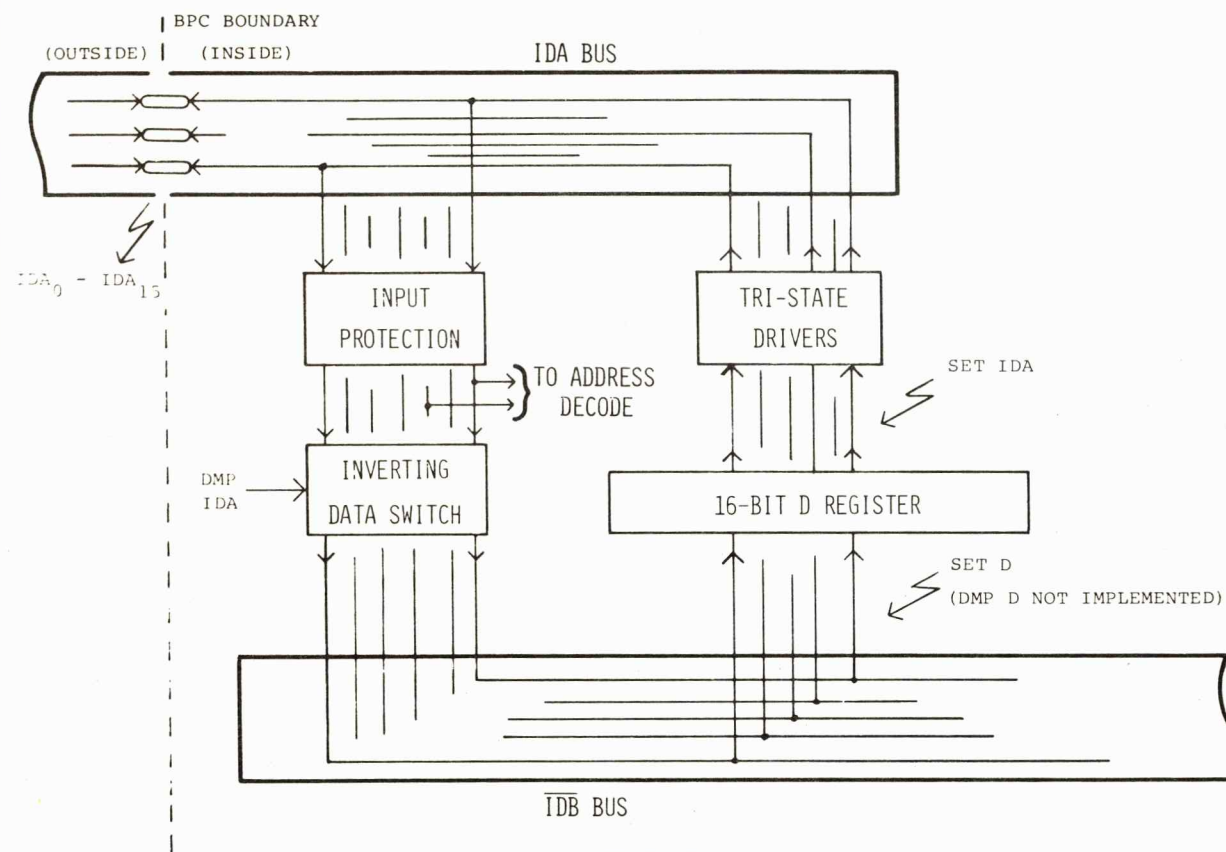


FIG 2-1

SECTION 2

Figure 2-1 is an overview of the connection between the external IDA Bus and the internal IDB Bus within the BPC.

The BPC accomplishes input of information from the IDA Bus by decoding a DMP IDA micro-instruction in a conjunction with some SET-register micro-instruction. The DMP IDA turns on the inverting Data Switch and the complement of the pattern on the IDA Bus is placed on the IDB Bus. The pattern on the IDA lines is always available to Address Decode. (Address Decode is responsible for recognizing memory cycles directed towards the BPC, and latching the address involved.)

To output a word of information onto the IDA Bus the BPC must first put the word of information into the D register. This is done with some DMP-register micro-instruction, in conjunction with a Set D. The only use of the D register is to drive the IDA Bus; therefore, there is no DMP D micro-instruction for putting D back onto the IDA Bus.

Once the data is in the D register a SET IDA micro-instruction will turn on the tri-state drivers and place the pattern on the IDA Bus. The tri-state drivers are devices that neither pull up nor pull down on the IDA Bus, unless a SET IDA is being given. This scheme allows other devices to drive the IDA Bus when the BPC is not driving the Bus.

Because of the large amount of capacitance possible on the IDA Bus, the BPC must use a SET D/SET IDA combination in one state, followed by a second SET IDA in another state, in order to guarantee proper information on the IDA Bus. This is done, for instance, during a memory cycle. For example, if the BPC is the originator of a memory cycle, when it sends the address it will delay the occurrence of Start Memory ($\overline{\text{STM}}$) so that it is coincident with the second SET IDA. In this way the BPC can guarantee that the address has been properly set up before the memory sees $\overline{\text{STM}}$.

DETAILS OF THE $\overline{\text{IDB}}$ BUS PRE-CHARGE, $\emptyset 1$ ENHANCER, AND DUMP START

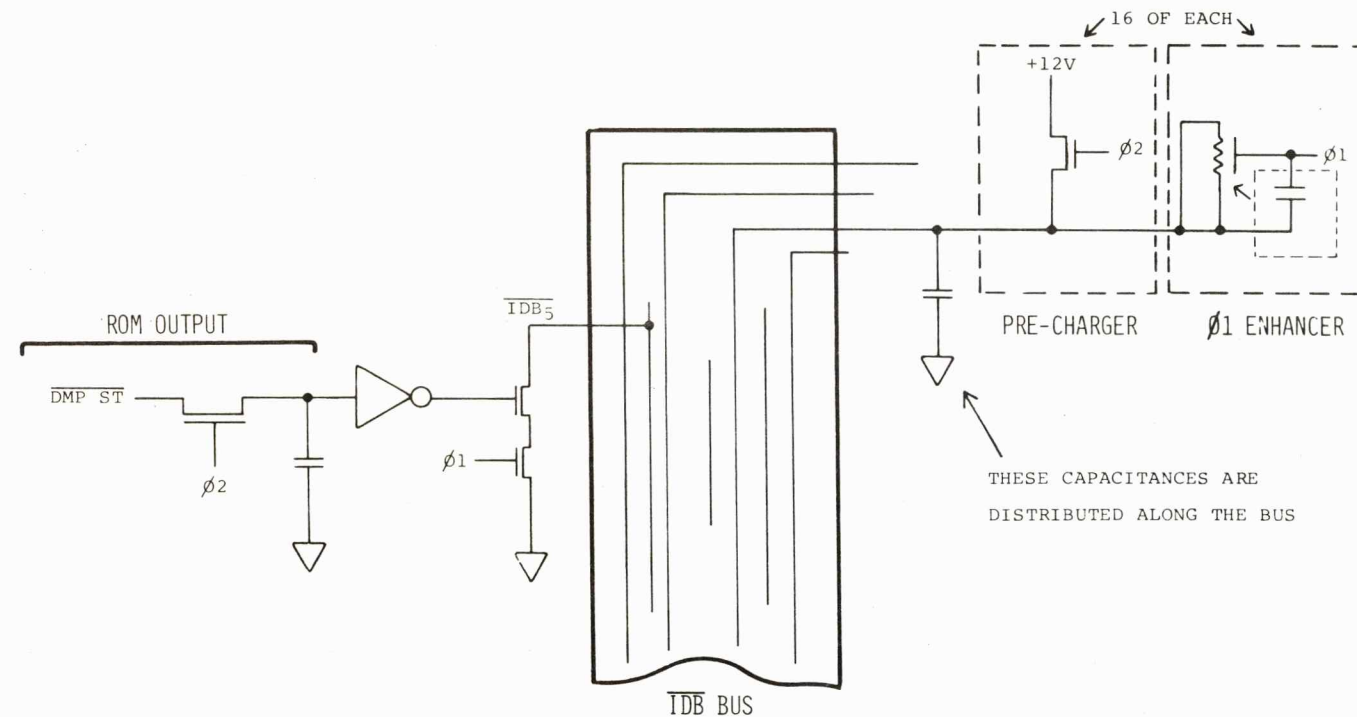


FIG 2-2

Figure 2-2 shows the circuitry that pre-charges the internal IDB Bus and the manner in which the DMP START micro-instruction is used.

During phase two each line of the IDB Bus is connected to +12V. This charges the distributed capacitance along each line. Micro-instructions that put information onto the Bus do so during phase one. What they do during phase one is either leave the charge on the line alone or ground it out. A micro-instruction taking data from the IDB Bus will do so at the end of phase one, after any needed discharges have been accomplished. Information exchange via the IDB Bus then, is accomplished via a phase two precharge followed by a subsequent discharging of various IDB lines on the next phase one.

The 01 Enhancer shown in the figure is a circuit which induces a greater electrical difference between high and low levels of individual lines on the IDB Bus. It does this by exploiting the thresh-

hold properties of transfer gates, in conjunction with capacitive coupling of phase one into the $\overline{\text{IDB}}$ Bus.

After the phase two pre-charge, each line having a high level is kept from sagging by the capacitive coupling of phase one into that line. This enhances an electrical high on that line. At the end of phase one a negative spike develops on the line, but phase one is removed from the receiving transfer gate at the very start of the spike, thereby shutting the gate off quickly enough to miss the spike.

After the phase two pre-charge, a low level on an IDB line is attained by some dump micro-instruction clamping the line down. This gets rid of the phase two precharge as well as the boost during phase one. But at the end of phase one the negative spike *is* coupled through to the receiving transfer gate. That gate does not turn off as quickly as in the previ-

OVERVIEW OF THE I REGISTER

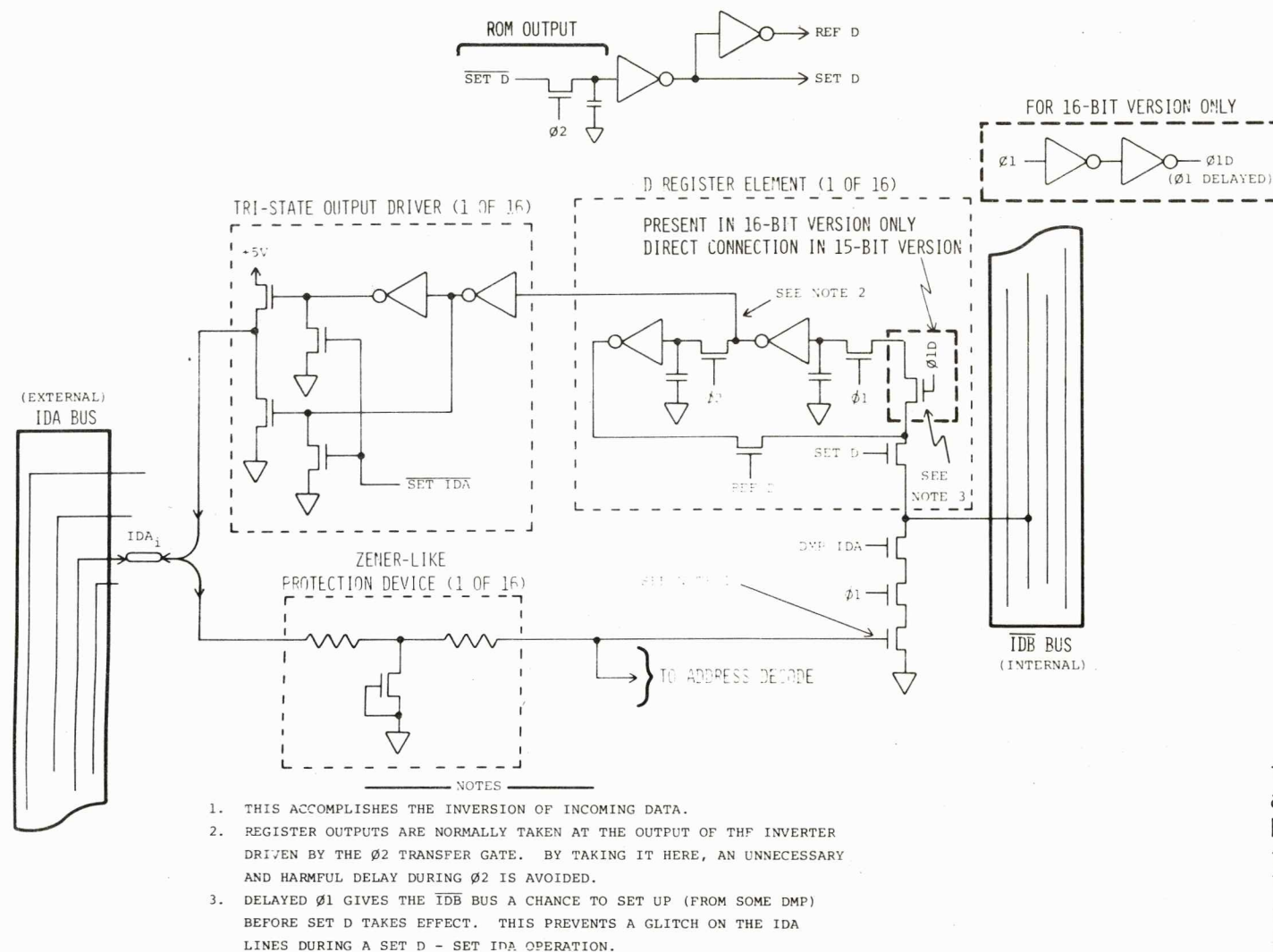


FIG 2-3

SECTION 2 (CONTINUED)

ous case, due to a lower difference in the gate/source voltage. This lower difference requires the gate terminal on the transfer gate transistor (phase one) to drop further. Thus the negative spike enhances an electrical low on a \overline{IDB} line.

The Phase One Enhancer mechanism is protected by a patent issued to Hewlett-Packard.

The purpose of the DMP START micro-instruction is to create a bit pattern on the IDB Bus representing octal 40. That pattern is used as an address from which to fetch the first instruction following initial turn-on.

Figure 2-3 depicts the details of the interconnection between the external IDA Bus and the internal IDB Bus. Input to the chip is accomplished by a DMP IDA micro-instruction. The mechanism is arranged so that the incoming data is inverted before it is put onto the IDB Bus. The protection devices are intended primarily to protect against static electricity.

Output from the BPC is accomplished by a SET D/SET IDA followed by a second SET IDA. Note that the output from the D register is available to the tri-state drivers during phase one. This is a half-state sooner than most register outputs are available to

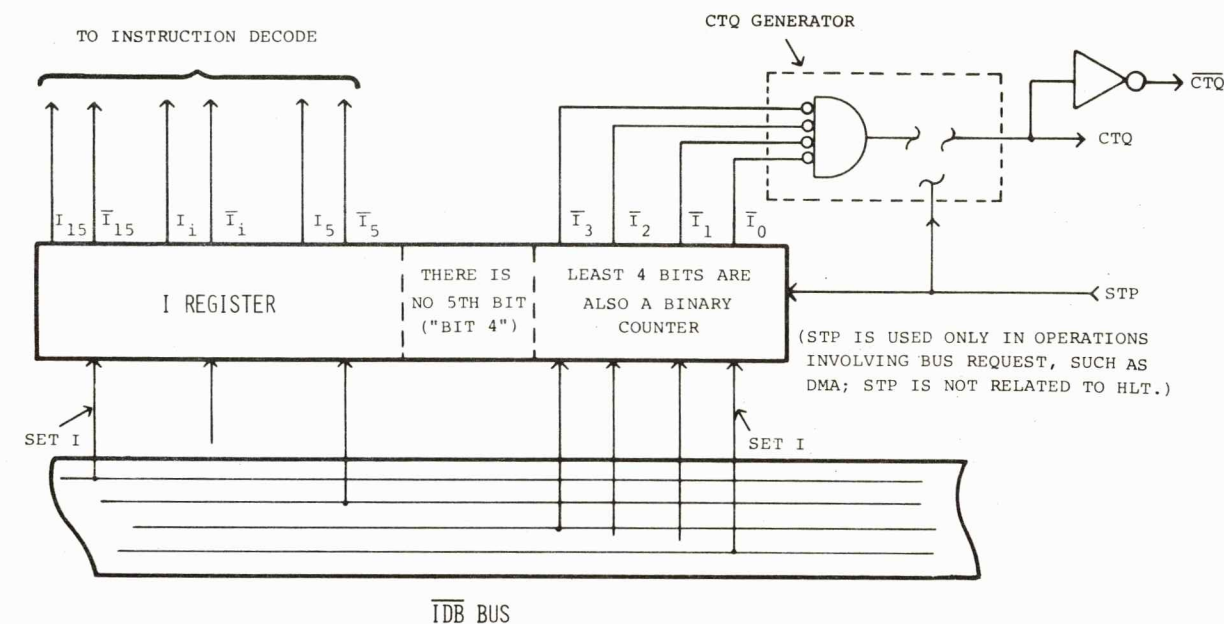


FIG 3-1

SECTION 3

Figure 3-1 is an overview of the I register. The purpose of the I register is to serve as a repository for the bit pattern obtained during an instruction fetch. The BPC machine-instruction bit patterns do not involve bit 4; therefore the I register does not contain a cell corresponding to that bit. Also, the contents of the I register are examined by dedicated circuitry that is continually connected to the I register. Therefore, since there is never any need to dump the I register, there is no DMP I in the micro-instruction. The upper eleven bits of the I register go directly to Instruction Decode. It is there that it is decided whether the bit pattern in the I register corresponds to a BPC instruction, and if so, what instruction.

Some BPC machine-instructions, such as the shift instructions, use their least four bits to encode a variable count of some sort. The initial value of that count is determined by the systems programmer at the time he generates the code, and is placed into the instruction bit pattern by the as-

OVERVIEW OF INSTRUCTION DECODE

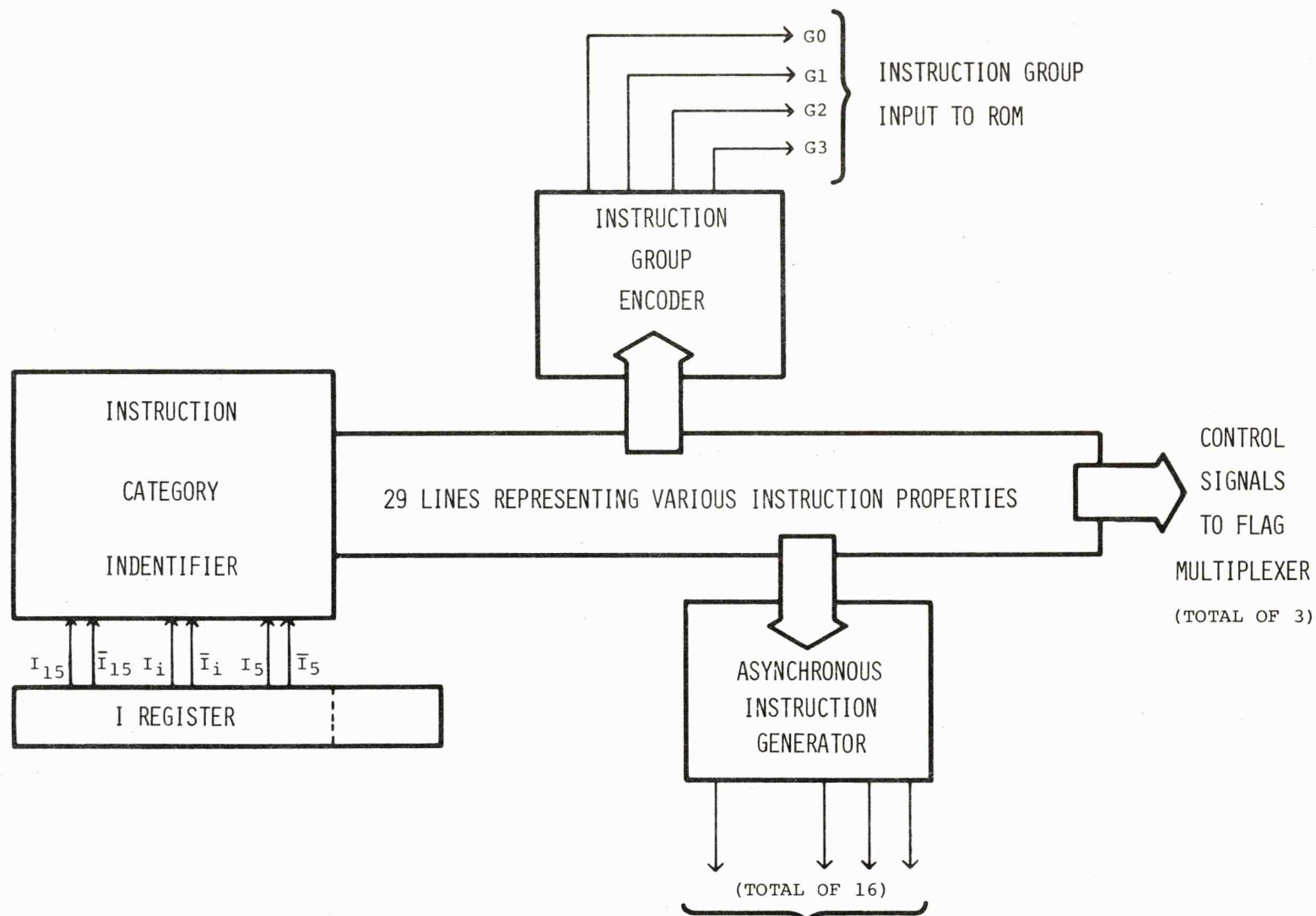


FIG 4-1

SECTION 4

Figure 4-1 is an overview of the Instruction Decode process. The I register supplies the basic instruction bit pattern to the Instruction Category Identifier. The Instruction Category Identifier is a complex series of AND gates that respond to various combinations of instruction bits. Each AND gate represents some property of a machine-instruction. The AND gates themselves are constructed from series of NOT'ed-input NOR gates. Their circuitry is not depicted. The Instruction Category Identifier generates 29 lines of output representing various instruction properties. These 29 lines are subject to further encoding and de-

coding, the results of which will control the subsequent actions of the BPC as it executes the machine-instruction.

The Instruction Group Encoder, for instance, controls the pattern of four qualifier signals G0-G3, so as to represent the major category within which the machine-instruction falls. These major categories differentiate between machine instructions which do load operations, store operations or shift operations, etc. By setting up the instruction group qualifiers the Instruction Group Encoder determines which branch of the ASM chart the BPC will follow during its execution of the machine-

instruction.

The Asynchronous Instruction Generator generates 16 steady state signals which are used to control the operation of various entities within the BPC. The asynchronous

TABLE OF INSTRUCTION CATEGORIES

SIGNAL NAME	MEANING AND ASSOCIATED ASSEMBLY LANGUAGE INSTRUCTIONS	FORMULA FOR GENERATION
LD*	INSTRUCTION IS EITHER LDA OR LDB (LOAD A OR LOAD B). $\overline{I_{11}}$ DIFFERENTIATES BETWEEN THE TWO.	$\overline{I_{14}} \cdot \overline{I_{13}} \cdot \overline{I_{12}}$
ST*	INSTRUCTION IS EITHER STA OR STB (STORE A OR STORE B). $\overline{I_{11}}$ DIFFERENTIATES BETWEEN THE TWO.	$\overline{I_{14}} \cdot \overline{I_{13}} \cdot I_{12}$
AD*	INSTRUCTION IS EITHER ADA OR ADB (ADD TO A OR ADD TO B). $\overline{I_{11}}$ DIFFERENTIATES BETWEEN THE TWO.	$\overline{I_{14}} \cdot \overline{I_{13}} \cdot \overline{I_{12}}$
CP*	INSTRUCTION IS EITHER CPA OR CPB (COMPARE TO A OR COMPARE TO B). $\overline{I_{11}}$ DIFFERENTIATES BETWEEN THE TWO.	$\overline{I_{14}} \cdot \overline{I_{13}} \cdot I_{12}$
$\overline{I_{11}}$	DIFFERENTIATES BETWEEN INSTRUCTIONS INVOLVING THE A REGISTER AND THOSE INVOLVING THE B REGISTER.	$\overline{I_{11}}$
JMP	INSTRUCTION IS JMP (JUMP).	$I_{14} \cdot \overline{I_{13}} \cdot \overline{I_{12}} \cdot I_{11}$
JSM	INSTRUCTION IS JSM (JUMP TO SUBROUTINE).	$I_{14} \cdot \overline{I_{13}} \cdot \overline{I_{12}} \cdot \overline{I_{11}}$
ISZ	INSTRUCTION IS ISZ (INCREMENT AND THEN SKIP IF ZERO).	$I_{14} \cdot \overline{I_{13}} \cdot \overline{I_{12}} \cdot I_{11}$
DSZ	INSTRUCTION IS DSZ (DECREMENT AND THEN SKIP IF ZERO).	$I_{14} \cdot \overline{I_{13}} \cdot I_{12} \cdot I_{11}$
AND	INSTRUCTION IS AND (AND TO A).	$I_{14} \cdot \overline{I_{13}} \cdot I_{12} \cdot \overline{I_{11}}$
IOR	INSTRUCTION IS IOR (INCLUSIVE OR WITH A).	$I_{14} \cdot \overline{I_{13}} \cdot \overline{I_{12}} \cdot \overline{I_{11}}$
ASG	INSTRUCTION IS AND ALTER-SKIP GROUP INSTRUCTION: RLA SKIP IF $A_0=1$, ALTER A_0 . RLB SKIP IF $B_0=1$, ALTER B_0 . SLA SKIP IF $A_0=0$, ALTER A_0 . SLB SKIP IF $B_0=0$, ALTER B_0 . SAP SKIP IF $A_{15}=0$, ALTER A_{15} . SBP SKIP IF $B_{15}=0$, ALTER B_{15} . SAM SKIP IF $A_{15}=1$, ALTER A_{15} . SBM SKIP IF $B_{15}=1$, ALTER B_{15} .	$I_{14} \cdot \overline{I_{13}} \cdot I_{12} \cdot I_{10}$

FIG 4-2-1

instructions are *not* similar to the micro-instructions which are issued from the ROM. Instead, the asynchronous instructions typically are used to determine the mode in which some device will behave when

INSTRUCTION CATEGORIES

INSTRUCTION DECODE OVERVIEW

CTQ

I REGISTER, SET I

I REGISTER COUNTER

TABLE OF INSTRUCTION CATEGORIES, CONTINUED

SIGNAL NAME	MEANING AND ASSOCIATED ASSEMBLY LANGUAGE INSTRUCTIONS	FORMULA FOR GENERATION
ASG (CONT.)	SOC SKIP IF OVERFLOW=0, ALTER OVERFLOW. SOS SKIP IF OVERFLOW=1, ALTER OVERFLOW. SEC SKIP IF EXTEND=0, ALTER EXTEND. SES SKIP IF EXTEND=1, ALTER EXTEND. RZA SKIP IF A IS NOT ZERO. RZB SKIP IF B IS NOT ZERO. SZA SKIP IF A IS ZERO. SZB SKIP IF B IS ZERO. RIA RZA, THEN INCREMENT A. RIB RZB, THEN INCREMENT B. SIA SZA, THEN INCREMENT A. SIB SZB, THEN INCREMENT B. SFS SKIP IF FLAG SET. SFC SKIP IF FLAG CLEAR. SSS SKIP IF STATUS SET. SSC SKIP IF STATUS CLEAR. SDS SKIP IF DECIMAL CARRY SET. SDC SKIP IF DECIMAL CARRY CLEAR. SHS SKIP IF HALT SET. SHC SKIP IF HALT CLEAR.	
I7	DIFFERNETIATES BETWEEN HOLD, AND ALTER BY SETTING OR CLEARING, THE TESTED BIT IN THE FOLLOWING ALTER INSTRUCTIONS: RLA, RLB, SLA, SLB, SAP, SBP, SAM, SBM, SOC, SOS, SEC, SES. I7=0 IMPLIES HOLD; I7=1 IMPLIES ALTER.	I_7
$\overline{I6}$	IF I7=0, $\overline{I6}$ DIFFERENTIATES BETWEEN CLEARING OR SETTING THE TESTED BIT, WHEN DOING AN ALTER INSTRUCTION. $\overline{I6}$ =1 IMPLIES CLEAR; $\overline{I6}$ =0 IMPLIES SET.	$\overline{I_6}$
LSC	INSTRUCTION IS AN ALTER INSTRUCTION INVOLVING A LEAST-SIGNIFICANT BIT: RLA, RLB, SLA, SLB.	$\overline{I_{15}} \cdot \overline{I_{14}} \cdot \overline{I_{13}} \cdot \overline{I_{12}} \cdot \overline{I_{10}} \cdot \overline{I_9}$
MSC	INSTRUCTION IS AN ALTER INSTRUCTION INVOLVING A MOST-SIGNIFICANT BIT, EXTEND, OR OVERFLOW: SAP, SAM, SBP, SBM, SOC, SOS, SEC, SES.	$I_{15} \cdot I_{14} \cdot I_{13} \cdot I_{12} \cdot I_{10}$

FIG 4-2-2

SECTION 4 (CONTINUED)

it receives certain micro-instructions from the ROM. For instance, an asynchronous control instruction might specify whether the S register is to shift left or shift right upon receipt of a shift micro-instruction from the ROM. Asynchronous

control instructions also determine the mode in which the Adder within the ALU operates.
The control signals to the Flag Multiplexer are similar in nature to the asynchronous instructions, and will be discussed at a

TABLE OF INSTRUCTION CATEGORIES, CONTINUED

SIGNAL NAME	MEANING AND ASSOCIATED ASSEMBLY LANGUAGE INSTRUCTIONS	FORMULA FOR GENERATION
RSS	DIFFERENTIATES BETWEEN THE SKIP AND REVERSE-SKIP SENSES AMONG THE ALTER AND SKIP GROUP INSTRUCTIONS.	$I_{14} \cdot I_{13} \cdot I_{12} \cdot I_{10} \cdot I_8$
EQ	INSTRUCTION IS EITHER SEC OR SES. RSS DIFFERENTIATES BETWEEN THE TWO.	$I_{15} \cdot I_{14} \cdot I_{13} \cdot I_{12} \cdot \overline{I_{11}} \cdot \overline{I_{10}} \cdot \overline{I_9}$
OQ	INSTRUCTION IS EITHER SOC OR SOS. RSS DIFFERENTIATES BETWEEN THE TWO.	$I_{15} \cdot I_{14} \cdot I_{13} \cdot I_{12} \cdot \overline{I_{11}} \cdot \overline{I_{10}} \cdot I_9$
S/RI	INSTRUCTIONS IS A "SKIP/REVERSE-SKIP, THEN INCREMENT" INSTRUCTION: SIA, RIA, SIB, RIB.	$\overline{I_{15}} \cdot \overline{I_{14}} \cdot \overline{I_{13}} \cdot \overline{I_{12}} \cdot I_{10} \cdot \overline{I_9} \cdot \overline{I_7} \cdot \overline{I_6}$
S/RZ	INSTRUCTION IS A "SKIP/REVERSE-SKIP, WITHOUT INCREMENT" INSTRUCTION: SZA, RZA, SZB, RZB.	$\overline{I_{15}} \cdot \overline{I_{14}} \cdot \overline{I_{13}} \cdot \overline{I_{12}} \cdot I_{10} \cdot \overline{I_9} \cdot \overline{I_7} \cdot \overline{I_6}$
FGC	IF ASG IS A 1, AND EACH OF S/RI, S/RZ, MSC AND LSC ARE 0'S, INSTRUCTION IS EITHER SFS OR SFC. RSS DIFFERENTIATES BETWEEN THE TWO.	$\overline{I_{11}} \cdot \overline{I_6}$
STC	IF ASG IS A 1, AND EACH OF S/RI, S/RZ, MSC AND LSC ARE 0'S, INSTRUCTION IS EITHER SSS OR SSC. RSS DIFFERENTIATES BETWEEN THE TWO.	$I_{11} \cdot \overline{I_6}$
HTC	IF ASG IS A 1, AND EACH OF S/RI, S/RZ, MSC AND LSC ARE 0'S, INSTRUCTION IS EITHER SHS OR SHC. RSS DIFFERENTIATES BETWEEN THE TWO. SEE NOTE AT BOTTOM.	$I_{11} \cdot I_6$
RET	INSTRUCTION IS RET (RETURN). [THE P/P BIT (I_6) IS A DON'T CARE BIT FOR THE BINARY PROCESSOR CHIP.]	$\overline{I_{15}} \cdot \overline{I_{14}} \cdot \overline{I_{13}} \cdot \overline{I_{12}} \cdot \overline{I_{11}} \cdot \overline{I_{10}} \cdot \overline{I_9} \cdot \overline{I_8} \cdot \overline{I_7}$
EXE	INSTRUCTION IS EXE (EXECUTE).	$I_{14} \cdot I_{13} \cdot I_{12} \cdot \overline{I_{11}} \cdot \overline{I_{10}} \cdot \overline{I_9} \cdot \overline{I_8} \cdot \overline{I_7} \cdot \overline{I_6} \cdot \overline{I_5}$
NOTE: THE NOR OF FGC, STC AND HTC, IF TRUE WHILE ASG IS ALSO TRUE, WHILE S/RI, S/RZ, MSC AND LSC ARE FALSE, SPECIFIES EITHER SDS OR SDC. RSS DIFFERENTIATES BETWEEN THE TWO.		

FIG 4-2-3

later time in connection with the Flag Multiplexer.

Figure 4-2 is a table indicating the nature of the 29 different instruction-related categories produced by the Instruction Category Identifier.

TABLE OF INSTRUCTION CATEGORIES, CONTINUED

SIGNAL NAME	MEANING AND ASSOCIATED ASSEMBLY LANGUAGE INSTRUCTIONS	FORMULA FOR GENERATION
T/CM	INSTRUCTION IS ANY COMPLEMENT INSTRUCTION: CMA, CMB, TCA, TCB (ONE'S COMPLEMENT OF A, ONE'S COMPLEMENT OF B, TWO'S COMPLEMENT OF A, TWO' COMPLEMENT OF B).	$I_{15} \cdot I_{14} \cdot I_{13} \cdot I_{12} \cdot \bar{I}_{10} \cdot \bar{I}_9 \cdot \bar{I}_8 \cdot \bar{I}_7 \cdot I_5$
TC*	INSTRUCTION IS EITHER TCA OR TCB (TWO'S COMPLEMENT A OR TWO'S COMPLEMENT B).	$I_{15} \cdot I_{14} \cdot I_{13} \cdot I_{12} \cdot \bar{I}_{10} \cdot \bar{I}_9 \cdot \bar{I}_8 \cdot \bar{I}_7 \cdot \bar{I}_6 \cdot I_5$
SRG	SHIFT-ROTATE GROUP—INSTRUCTION IS ONE OF THE FOLLOWING: AAR ARITHMETIC RIGHT SHIFT OF A. ABR ARITHMETIC RIGHT SHIFT OF B. SAR SHIFT A RIGHT. SBR SHIFT B RIGHT. SAL SHIFT A LEFT. SBL SHIFT B LEFT. RAR ROTATE A RIGHT. RBR ROTATE B RIGHT. \bar{I}_6 , I_7 AND \bar{I}_{11} ARE USED TO DISTINGUISH BETWEEN THESE INSTRUCTIONS.	$I_{15} \cdot I_{14} \cdot I_{13} \cdot I_{12} \cdot \bar{I}_{10} \cdot \bar{I}_9 \cdot I_8 \cdot \bar{I}_5$

FIG 4-2-4

TABLE OF INSTRUCTION BIT PATTERNS

GROUP: MEMORY REFERENCE

INST.

111 DIFFERENTIATES BETWEEN A&B FOR THESE INSTRUCTIONS

NAME

		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LD*	LDA	D/I	0	0	0	0	B/ \bar{B}										
	LDB	D/I	0	0	0	1	B/ \bar{B}										
CP*	CPA	D/I	0	0	1	0	B/ \bar{B}										
	CPB	D/I	0	0	1	1	B/ \bar{B}										
AD*	ADA	D/I	0	1	0	0	B/ \bar{B}										
	ADB	D/I	0	1	0	1	B/ \bar{B}										
ST*	STA	D/I	0	1	1	0	B/ \bar{B}										
	STB	D/I	0	1	1	1	B/ \bar{B}										
JSM	JSM	D/I	1	0	0	0	B/ \bar{B}										
ISZ	ISZ	D/I	1	0	0	1	B/ \bar{B}										
AND	AND	D/I	1	0	1	0	B/ \bar{B}										
DSZ	DSZ	D/I	1	0	1	1	B/ \bar{B}										
IOR	IOR	D/I	1	1	0	0	B/ \bar{B}										
JMP	JMP	D/I	1	1	0	1	B/ \bar{B}										

- * 10 BIT ADDRESS FIELD.
- * ADDRESSES 0-37₈ ARE REGISTERS.
- * FOR BIT 9=0, BITS 0-8 = POSITIVE ADDR.
- * FOR BIT 9=1, ADDRESS IS NEGATIVE.
- IGNORE BIT 9, **COMPLEMENT** BITS 0-8, THEN ADD ONE.
- * BASE PAGE ADDRESS ENCODING IS ALWAYS WITH RESPECT TO MEMORY LOCATION ZERO.
- * CURRENT PAGE ENCODING:
 - (ABSOLUTE) RELATIVE TO THE MIDDLE OF THE PAGE (1000B, 3000B, ETC.)
 - (RELATIVE) RELATIVE TO THE CURRENT VALUE OF P, +511, -512.

NONE OF THESE INSTRUCTIONS HAVE 111 IN THESE BIT POSITIONS, ALL OTHER INSTRUCTIONS DO.

D/I (DIRECT/INDIRECT) AND B/ \bar{B} (BASE PAGE/ NOT BASE PAGE) ARE CODED AS 0/1.

FIG 4-3-1

SECTION 4 (CONTINUED)

Figure 4-3 illustrates the various instruction bit-patterns for BPC machine-instructions and their relationship to the 29 categories generated by the Instruction Category Identifier.

TABLE OF INSTRUCTION BIT PATTERNS, CONTINUED

GROUP: ALTER

PART OF INSTRUCTION CATEGORY IDENTIFIED BY ASG

RSS=ASG·I₈

INST. NAME

111

I7

I6

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
LSC {	RLA	0	1	1	1	0	1	1	1	H/H	C/S	* 6 BIT SKIP FIELD, +31, -32. * IF BIT 5=0, SKIP TO P+=; #=BITS 0 THRU 4. * IF BIT 5=1, SKIP TO P-=, #=1+ COMP OF BITS 0-4.									
	RLB	0	1	1	1	1	1	1	1	H/H	C/S										
	SLA	0	1	1	1	0	1	1	0	H/H	C/S										
	SLB	0	1	1	1	1	1	1	0	H/H	C/S										
MSC {	SAP	1	1	1	1	0	1	0	0	H/H	C/S										
	SBP	1	1	1	1	1	1	0	0	H/H	C/S										
	SAM	1	1	1	1	0	1	0	1	H/H	C/S										
	SBM	1	1	1	1	1	1	0	1	H/H	C/S										
	OQ {	SOC	1	1	1	1	0	1	1	0	H/H	C/S									
		SOS	1	1	1	1	0	1	1	1	H/H	C/S									
	EQ {	SEC	1	1	1	1	1	1	1	0	H/H	C/S									
		SES	1	1	1	1	1	1	1	1	H/H	C/S									

H/H (HOLD/DON'T HOLD) AND C/S (CLEAR/SET) ARE CODED AS 0/1.

GROUP: SKIP

REMAINDER OF ASG CATEGORY INSTRUCTIONS

INST. NAME

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S/RZ { RZA	0	1	1	1	0	1	0	0	0	0	* 6 BIT SKIP FIELD, +31, -32. * IF BIT 5=0, SKIP TO P+; #=BITS 0 THRU 4. * IF BIT 5=1, SKIP TO P-; #=1+ COMP OF BITS 0-4.					
RZB	0	1	1	1	1	1	0	0	0	0						
SZA	0	1	1	1	0	1	0	1	0	0						
SZB	0	1	1	1	1	1	0	1	0	0						
S/RI { RIA	0	1	1	1	0	1	0	0	0	1						
RIB	0	1	1	1	1	1	0	0	0	1						
SIA	0	1	1	1	0	1	0	1	0	1						
SIB	0	1	1	1	1	1	0	1	0	1						
FGC { SFS	0	1	1	1	0	1	0	0	1	0						
SFC	0	1	1	1	0	1	0	1	1	0						
STC { SSS	0	1	1	1	1	1	0	0	1	0						
SSC	0	1	1	1	1	1	0	1	1	0						
FGC.STC.HTC { SDS	0	1	1	1	0	1	0	0	1	1						
SDC	0	1	1	1	0	1	0	1	1	1						
HTC { SHS	0	1	1	1	1	1	0	0	1	1						
SHC	0	1	1	1	1	1	0	1	1	1						

FIG 4-3-2

TABLE OF INSTRUCTION BIT PATTERNS, CONTINUED

GROUP: COMPLEMENT

INST. NAME

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T/CM { CMA	1	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0
CMB	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0
TC* { TCA	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0
TCB	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0

GROUP: RETURN

INST. NAME

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET { RET	1	1	1	1	0	0	0	0	1	P/P	6 BIT, 2'S COMPLEMENT SKIP FIELD (ALLOWS -32 THRU +31).					

P/P (DON'T POP/POP THE IOC) ENCODED AS 0/1.

GROUP: EXECUTE

INST. NAME

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXE { EXE	D/I	1	1	1	0	0	0	0	0	0	0	5 BIT REGISTER ADDRESS (0-37 ₈).				

D/I (DIRECT/INDIRECT) ENCODED AS 0/1.

GROUP: SHIFT-ROTATE

INST. NAME

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRG { AAR	1	1	1	1	0	0	0	1	0	0	0	0	* 4 BITS OF SHIFT-ROTATE FIELD. * IN SOURCE 1 ≤ N ≤ 16. * BINARY IN THIS FIELD IS N-1.			
ABR	1	1	1	1	1	0	0	1	0	0	0	0				
SAR	1	1	1	1	0	0	0	1	0	1	0	0				
SBR	1	1	1	1	1	0	0	1	0	1	0	0				
SAL	1	1	1	1	0	0	0	1	1	0	0	0				
SBL	1	1	1	1	1	0	0	1	1	0	0	0				
RAR	1	1	1	1	0	0	0	1	1	1	0	0				
RBR	1	1	1	1	1	0	0	1	1	1	0	0				

FIG 4-3-3

TABLE OF GROUP
QUALIFIER ENCODING

INSTRUCTION CATEGORY	GROUP QUALIFIERS GENERATED			
	G3	G2	G1	G0
ST*	1	1	1	1
AD*	0	1	1	1
JMP	1	0	1	0
EXE	1	1	1	0
ASG	1	1	0	0
LD*	0	1	1	1
SRG	0	1	0	0
ISZ	0	0	1	0
JSM	1	0	1	1
DSZ	0	0	1	0
RET	1	0	0	0
T/CM	0	1	0	1
CP*	0	1	1	0
AND	0	1	1	1
IOR	0	1	1	1

FIG 4-4

SECTION 4 (CONTINUED)

Figure 4-4 illustrates how selected instructions categories can generate patterns of group qualifiers. The circuitry for this is not shown, but consists of a ROM like array of pull-down transistors driven by the instruction category lines.

Figure 4-5 illustrates the details of the Asynchronous Instruction Generator. The figure shows how the various instruction category lines are gated together and where the results are sent. The meaning and uses of each line that is generated are discussed in connection with the various circuits that utilize them.

DETAILS OF THE ASYNCHRONOUS INSTRUCTION GENERATOR

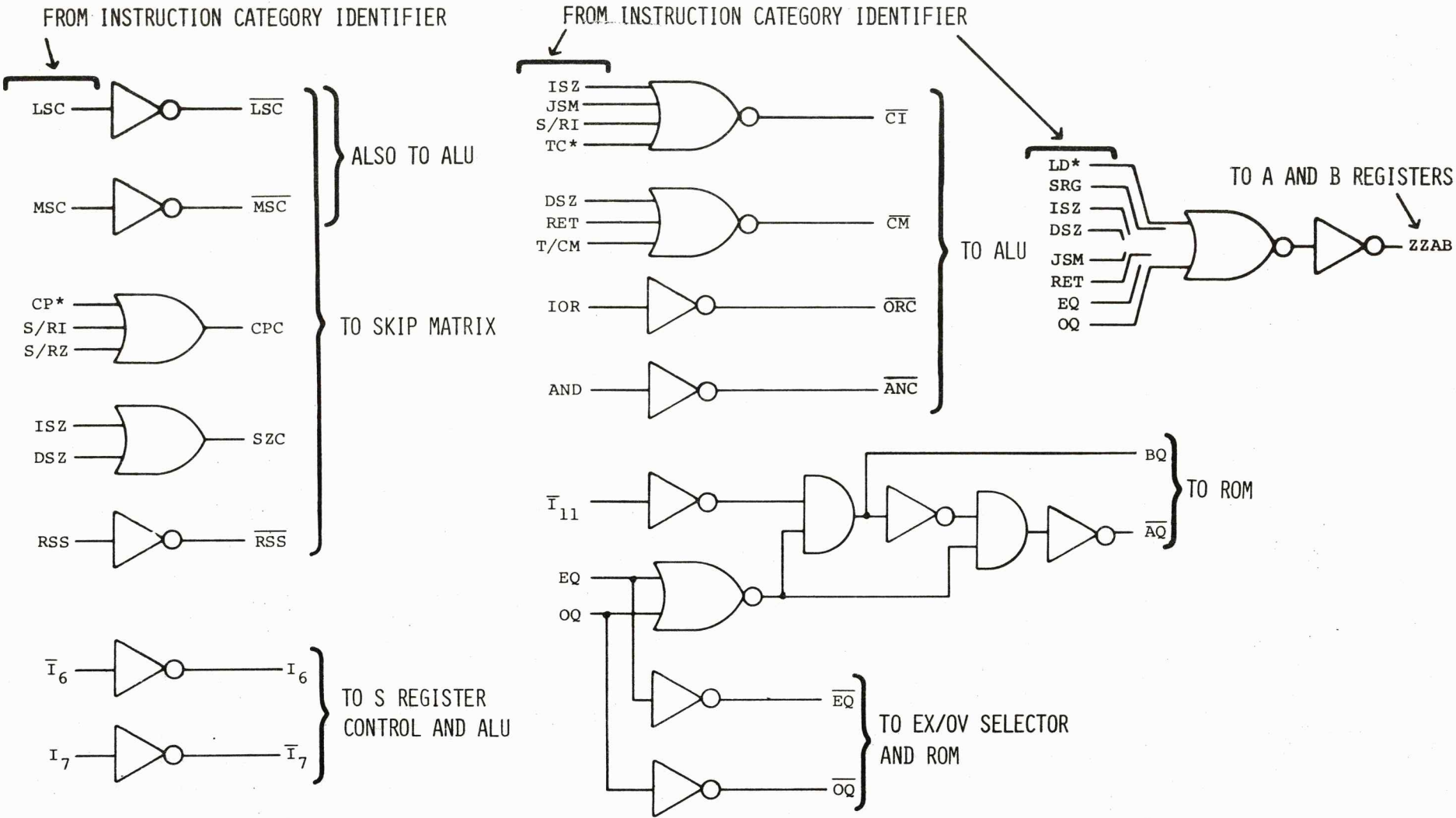


FIG 4-5

SECTION 5

Figure 5-1 is an overview of the ROM structure used by the state machine which executes the BPC machine-instructions. The ROM is actually organized as a program logic array (PLA). A PLA differs from an ordinary ROM in the following way. A regular ROM has addresses as inputs and instructions as outputs. Corresponding to each address is a unique combination of instructions; just what the combination is for each address is determined at the time the ROM is programmed. In an ordinary ROM there is an internal line that corresponds to each address that the ROM can respond to. Each time an address is given the corresponding

line is energized. That energized line then causes each of the instructions that that address is suppose to issue. Changing even one input address bit energizes an entirely different line and generally results in an entirely different combination of instructions being issued. In contrast to this, the PLA has qualifiers as input and instructions as output. Typically, a PLA has many more qualifiers as input than a ROM has address bits as input. Each instruction that can be decoded is responsible for decoding the qualifiers in order to determine when the instruction is to be issued. Often a certain micro-instruction is decoded under quite separate

INSTRUCTION
BIT PATTERNS (CONT)
INSTRUCTION
BIT PATTERNS (CONT)
GROUP QUALIFIER
ENCODING
ASYNCHRONOUS CONTROL
LINE GENERATION

OVERVIEW OF THE CONTROL ROM SYSTEM

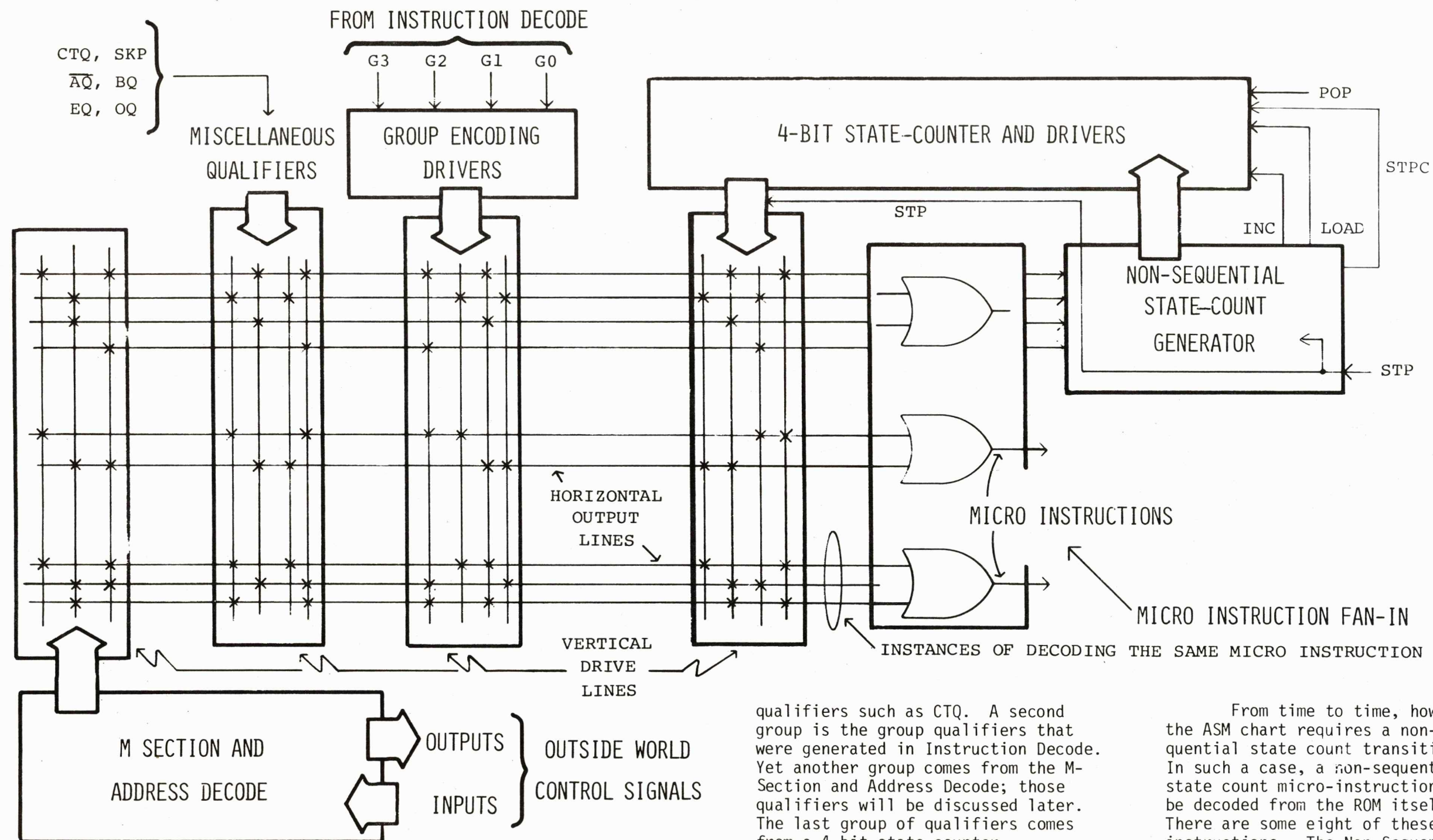


FIG 5-1

SECTION 5 (CONTINUED)

circumstances. In such a case the various instances of decoding must be OR'ed together to produce a single line which stands for the issuing of that micro-instruction. Changing a single input qualifier may result in no changes at all in the decoded micro-instructions, or, it may result in all of them changing.

It just depends on how the various micro-instructions respond with respect to that particular qualifier. Throughout the documentation for the processor we call a PLA a ROM and let it go at that.

The BPC's ROM has four general groups of input qualifiers. One group is composed of miscellaneous

qualifiers such as CTQ. A second group is the group qualifiers that were generated in Instruction Decode. Yet another group comes from the M-Section and Address Decode; those qualifiers will be discussed later. The last group of qualifiers comes from a 4-bit state counter.

The ASM chart for the BPC is divided into a number of branches. Each branch has a number of steps down its length. The group qualifiers select the branch; the State Counter determines which particular step in the branch is active. The State Counter automatically increments itself through its sequence of counts. The native sequence of the State Counter was chosen (by computer simulation) to minimize the number of non-sequential state count transitions required.

From time to time, however, the ASM chart requires a non-sequential state count transition. In such a case, a non-sequential state count micro-instruction will be decoded from the ROM itself. There are some eight of these instructions. The Non-Sequential State Count Generator responds to each of the eight non-sequential instructions by forcing the State Counter to its proper state count for each such instruction.

The following signals are also used in connection with the State Counter. POP forces the State Counter to a particular state which is never used for anything except start-up. INC merely means that no non-sequential state count instruction is in effect, and that the native transition should occur

DETAILS OF THE ROM STATE-COUNTER AND THE GROUP ENCODING DRIVERS

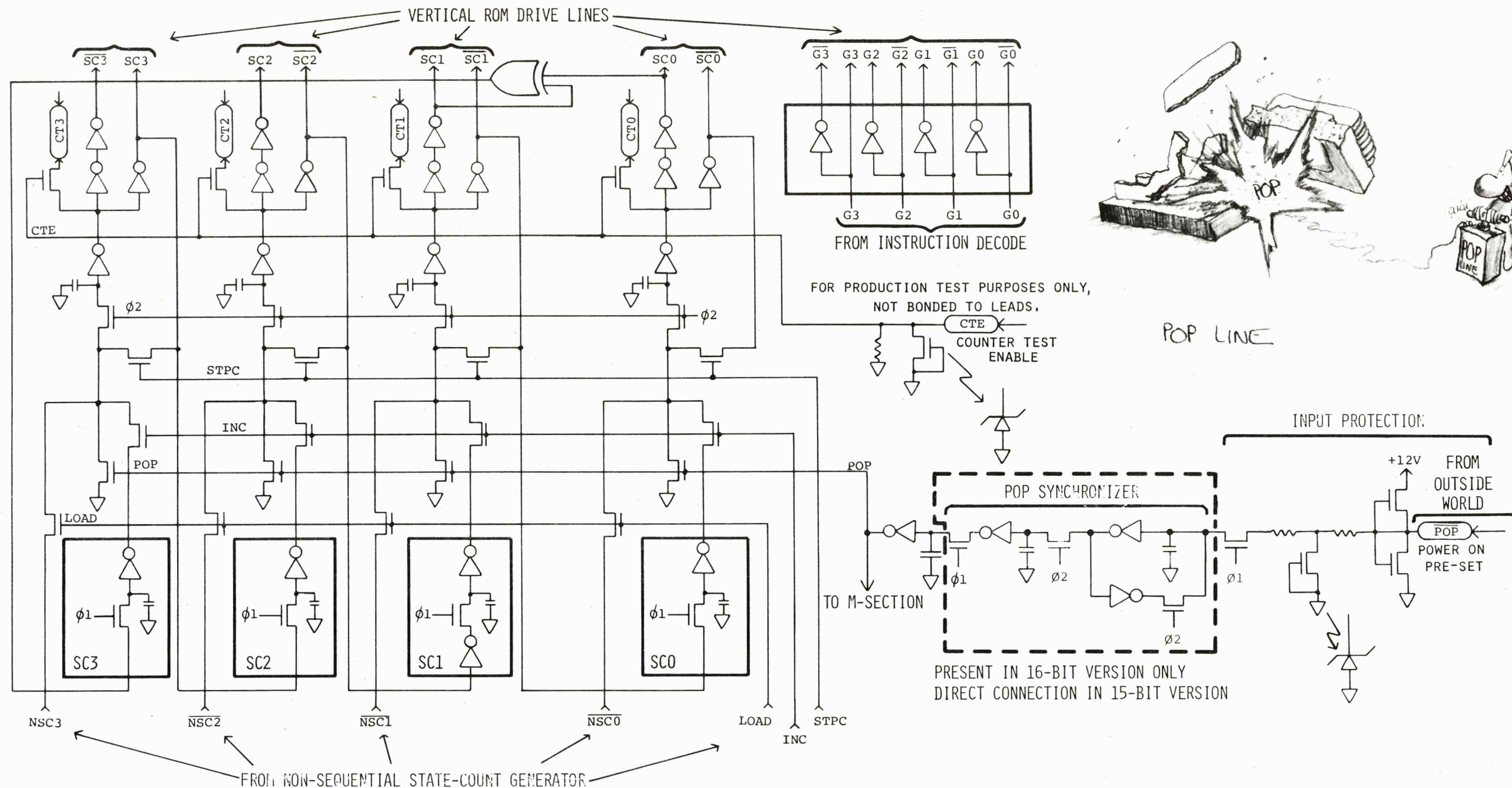


FIG 5-2-1

SECTION 5 (CONTINUED)

within the State Counter. LOAD means that a non-sequential state count instruction has been received, and that the native increment should be suspended and the State Counter set according to the patterns supplied by the Non-Sequential State Count Generator. STP is generated when, for any reason, a Bus Grant has occurred. It causes both INC and LOAD to go false, suspending

the operation of the State Counter. STP is also sent into the ROM as a qualifier. In this way the occurrence of an STP can suspend the decoding of most micro-instructions coming from the ROM.

Figure 5-2 illustrates the operation of the ROM State Counter. Once inside the ROM itself, the state count lines (SC0-SC3) are treated just as any other qualifier lines are. (The same is true of the group qualifiers from Instruction Decode.) The pattern for the State Counter is a simple right shift from bit 3 to bit 2 to bit 1. The complement of bit 1 is put into bit

0. The complement of the exclusive OR of bit 0 and bit 1 is shifted into bit 3. The resulting pattern of state counts and their assignment of state numbers is shown in the table.

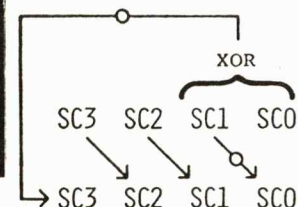
An important aspect of the State Counter's operation is its phasing. At the start of phase two the state count qualifier lines SC0-SC3 transition to their new

STATE-COUNTER, POP

CONTROL ROM
OVERVIEW

ROM STATE-COUNT PATTERN

RULE FOR COMPUTING NTH STATE COUNT FROM N-1TH STATE COUNT:



1. \rightarrow DENOTES NTH SC(I) = N-1TH SC(I-1)
2. \rightarrow DENOTES NTH SC(I) = N-1TH SC(I-1)

THESE STATES NOT USED

START-UP STATE
(CAUSED BY POP)

"FORBIDDEN" STATE
(TRANSITIONS TO
ITSELF ONLY)

ROM STATE-COUNTS

ASM CHART STATE #	ROM STATE COUNTER OUTPUTS			
	SC3	SC2	SC1	SC0
14	0	1	1	1
0	1	0	1	0
1	0	1	0	0
2	1	0	1	1
3	1	1	0	0
4	1	1	1	1
5	1	1	1	0
6	0	1	1	0
7	0	0	1	0
8	0	0	0	0
9	1	0	0	1
10	0	1	0	1
11	0	0	1	1
12	1	0	0	0
13	1	1	0	1
14	0	1	1	1

0	0	0	1
0	0	0	1

FIG 5-2-2

SECTION 5 (CONTINUED)

values. As the qualifier lines are setting-up, the various decode lines in the ROM are being precharged. All qualifier lines must be stable by the end of phase two. During the next phase one ROM outputs are decoded. The next state count is also clocked into the back half of the State Counters during the next phase two. It will become the state count unless a non-sequential state count (NSC) instruction was decoded from the ROM itself. During such an occurrence LOAD will go true and INC will go false. During the next phase two the NSC pattern generated by the instruction will be gated directly into the second half

of the state counters, bypassing the input created by the State Counter itself during the previous phase one.

Observe that STPC causes the second half of the State Counter to latch its output back into itself. This prevents the state count from changing. It is not necessary to latch the phase one portion of the State Counter, since it follows the now captive phase two portion through the state counter's native incrementing rule.

Also observe how POP forces the State Counter to a preselected state count. The differences between the 15 and 16-bit versions have to do with the way the POP is

DETAILS OF ROM OUTPUT DECODING

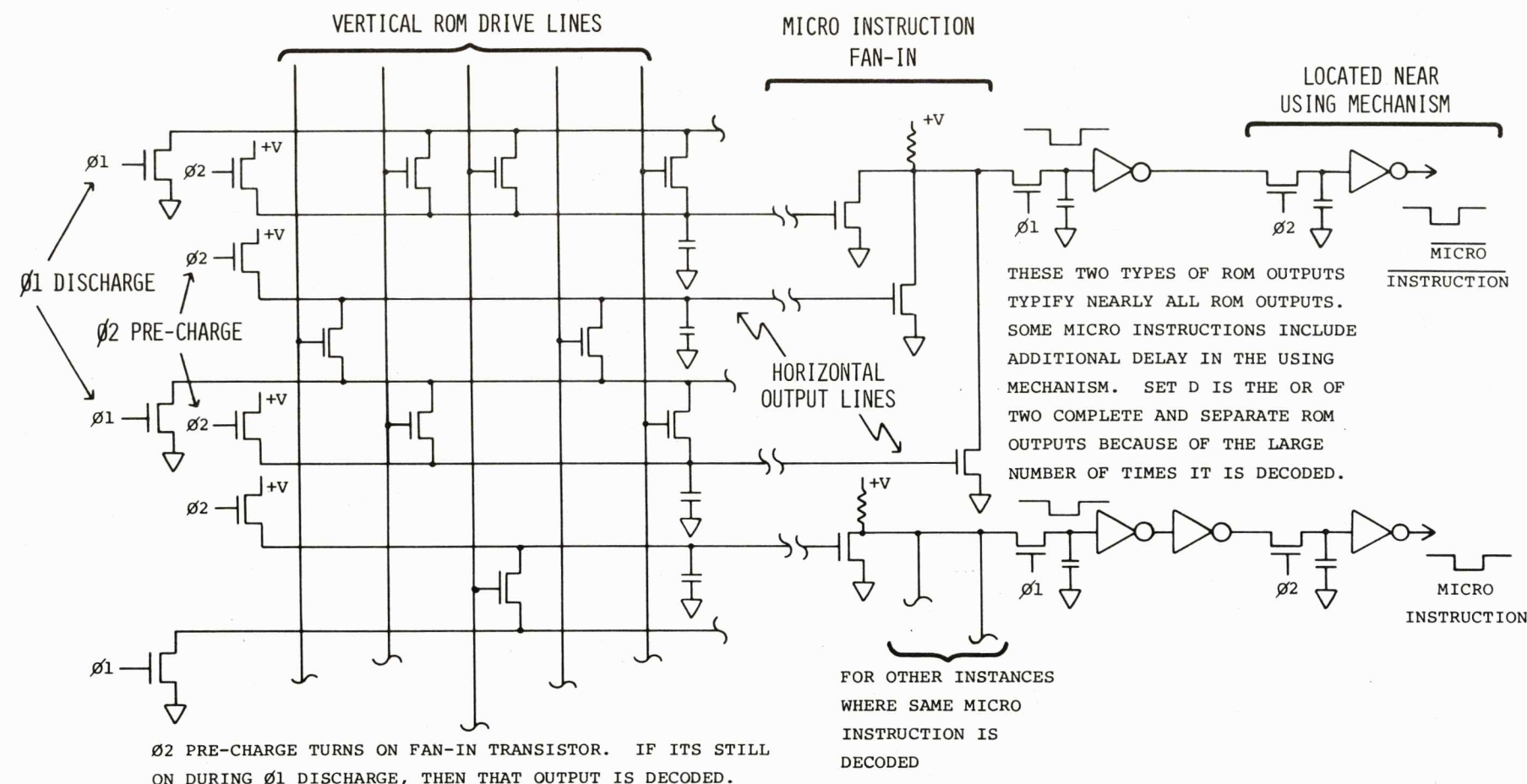


FIG 5-3

handled. In the 15-bit version POP must be released during phase two so that it is stable at the onset of phase one. The 16-bit version contains a POP Synchronizer which supposedly removes the need for such synchronization. At least one authority, however, claims that the synchronizer trick doesn't work.

Figure 5-3 illustrates the decoding of micro-instructions from the ROM. The vertical ROM drive lines consist of the state count lines, group qualifier lines, and other qualifier lines sent to the ROM. Each qualifier line is generally present in its true and complement sense. Each horizontal output line represents a particular micro-instruction to be decoded under certain circumstances. Each such horizontal output line is precharged

during phase two. Each pair of horizontal output lines is served by a common discharge bus that is grounded during phase one. To decode a micro-instruction, the decoding transistors are arranged such that when the conditions are met none of them turns on. This isolates the horizontal output line from the discharge bus, and the precharge remains into phase one. All of the horizontal output lines representative of a particular micro-instruction are collected together by a NOR gate, and the output is inverted, or not, and delayed until phase two.

If a horizontal output line is not selected, then at least one decoding transistor will connect that horizontal line to its discharge bus. In that case the phase two precharge is sucked out by the phase one discharge, and no output results.

OVERVIEW OF THE NON-SEQUENTIAL STATE-COUNT GENERATOR

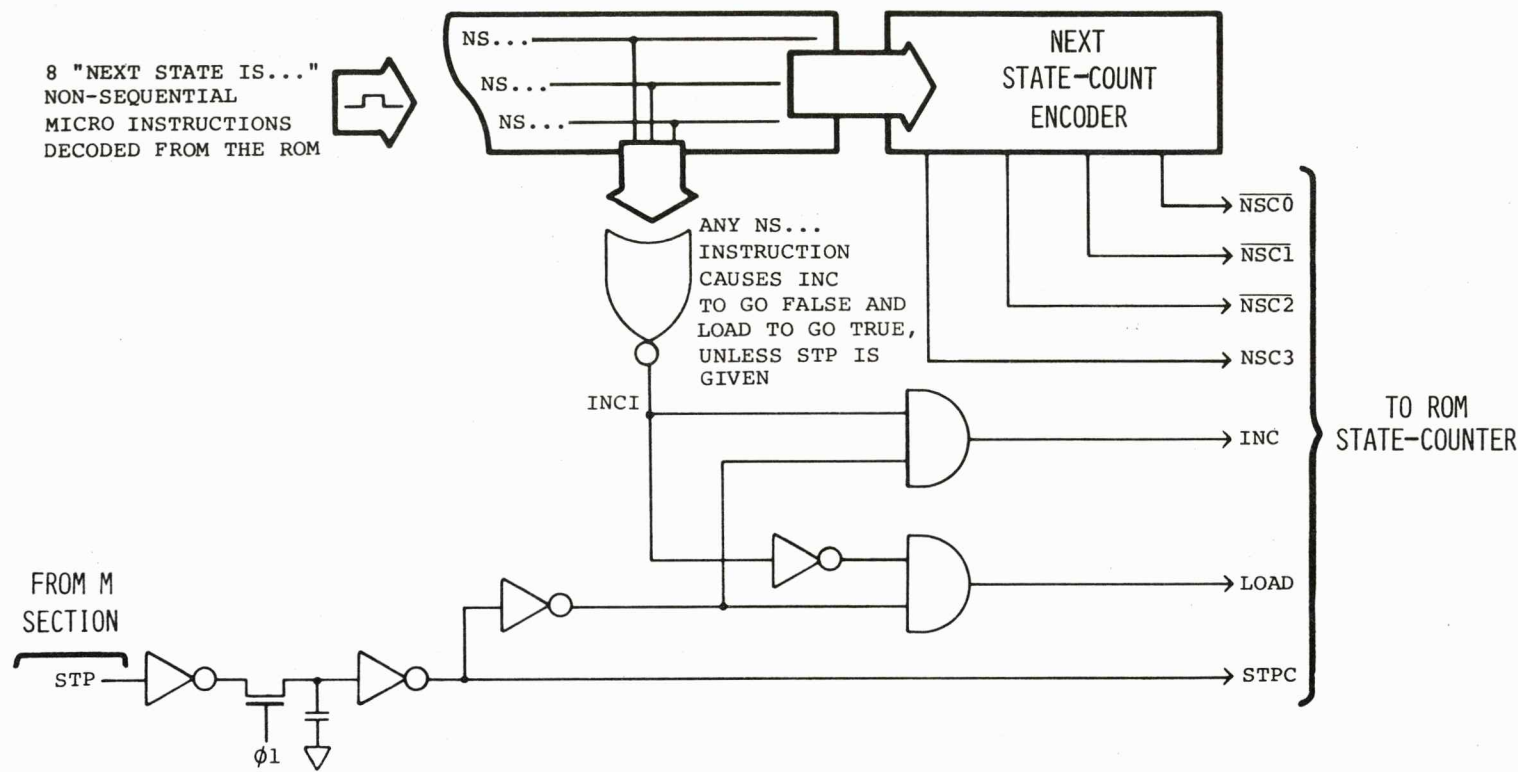


FIG 5-4-1

THE WHY AND WHEREFORE OF THE NON-SEQUENTIAL STATE-COUNT GENERATOR

THE ASM CHART HAS THESE 20 NON-SEQUENTIAL STATE-COUNT TRANSITIONS:

0 to 0	3 to 2	2 to 4	3 to 7	2 to 10
5 to 0	4 to 2	4 to 4	5 to 7	6 to 10
10 to 0	3 to 3	2 to 6	4 to 9	7 to 10
2 to 2	6 to 3	3 to 6	9 to 9	14 to 10

REQUIRING THESE 8 "NEXT-STATE IS..." NON-SEQUENTIAL STATE-COUNT MICRO-INSTRUCTIONS:

NS0	NS3	NS6	NS9
NS2	NS4	NS7	NS10

DEFINITION OF THE "NEXT-STATE IS..." NON-SEQUENTIAL MICRO-INSTRUCTIONS:

NON-SEQUENTIAL STATE-COUNT MICRO-INSTRUCTION	STATE-COUNTER CONTROL LINES				
	INCI	NSC3	NSC2	NSC1	NSC0
NS0	0	1	1	0	1
NS2	0	1	1	0	0
NS3	0	1	0	1	1
NS4	0	1	0	0	0
NS6	0	0	0	0	1
NS7	0	0	1	0	1
NS9	0	1	1	1	0
NS10	0	0	0	1	0
NONE OF THE ABOVE	1	-	-	-	-

← THESE PATTERNS, WHEN PUT THROUGH THE STATE-COUNTER'S RULE FOR STATE-TO-STATE TRANSITIONS, RESULT IN THE DESIRED STATE-COUNTS.

FIG 5-4-2

SECTION 5 (CONTINUED)

Figure 5-4-1 is an overview of the Non-Sequential State Count Generator. Each non-sequential next state micro-instruction causes the Next State Count Encoder to generate the next state count pattern. In addition, there is shown the means whereby the occurrence of any such micro-instruction causes INC to go true and LOAD to go false. Observe also that STP causes both INC and LOAD to go false.

Figure 5-4-2 summarizes the activity that the Next State Count Encoder must perform.

ACTUAL NSC TRANSITIONS

NSC GENERATOR

ROM OUTPUT DECODING

NATIVE STATE-COUNT

DETAILS OF THE NEXT STATE-COUNT ENCODER

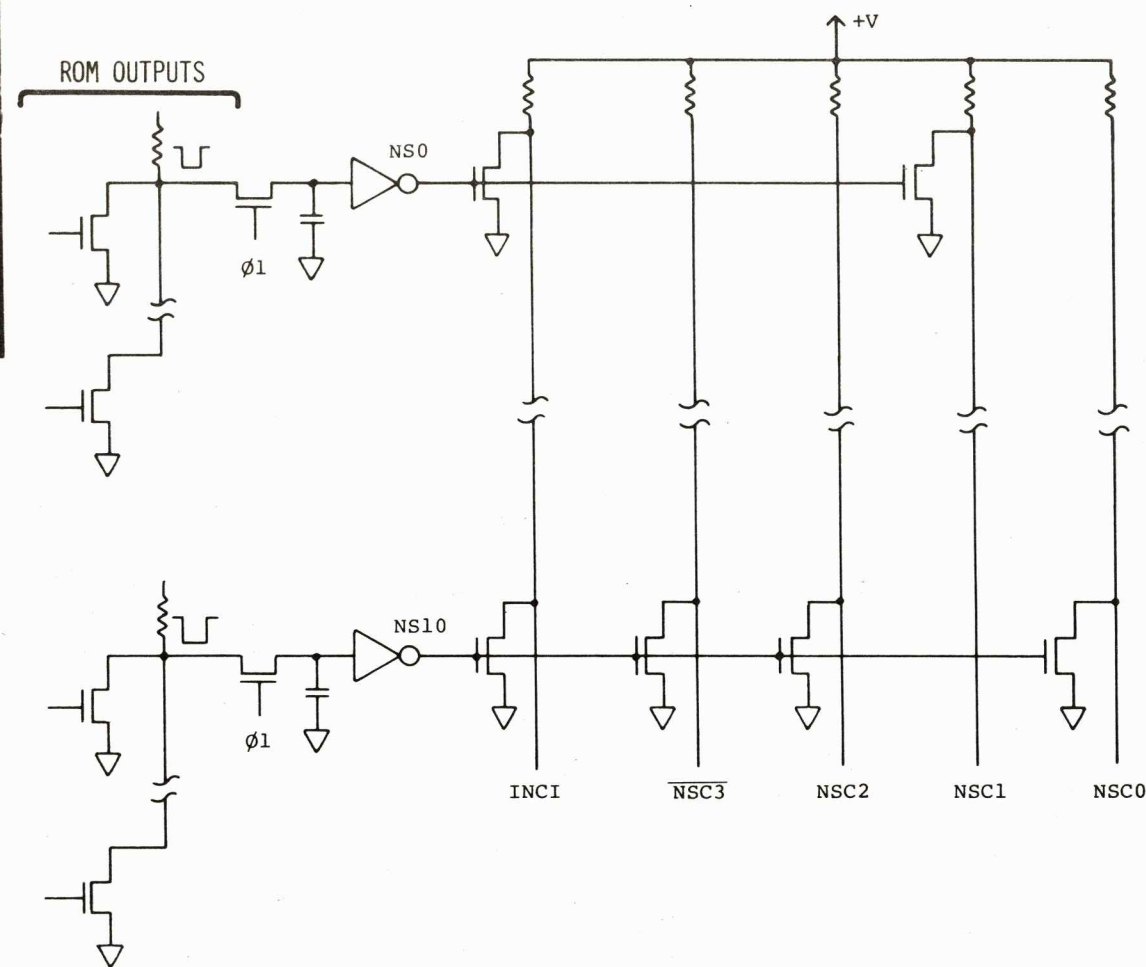


FIG 5-4-3

SECTION 5 (CONTINUED)

Figure 5-4-3 illustrates the actual circuitry of the Next State Count Encoder.

THE R REGISTER

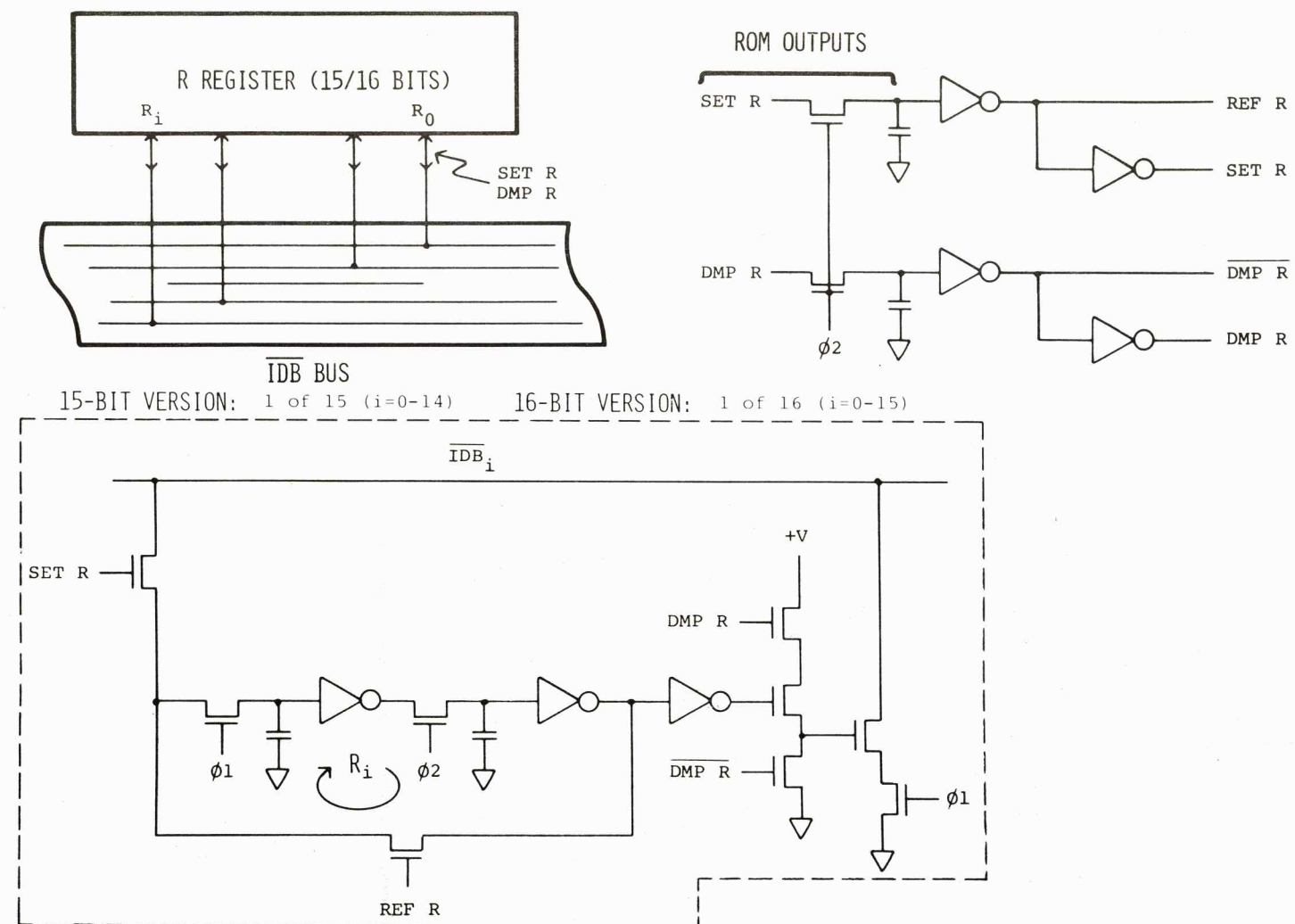
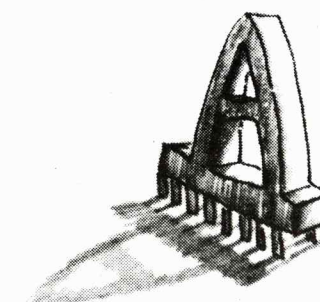


FIG 6-1

SECTION 6

Figure 6 illustrates the details of the R register. Although the R register is probably the simplest register in the BPC, it exhibits typical register architecture.

The difference between the 15-bit and 16-bit versions is simply that the 16-bit version has a full 16-bit R register, while the 15-bit version has merely a 15-bit R register.



THE "A" REGISTER

OVERVIEW OF THE A AND B REGISTERS

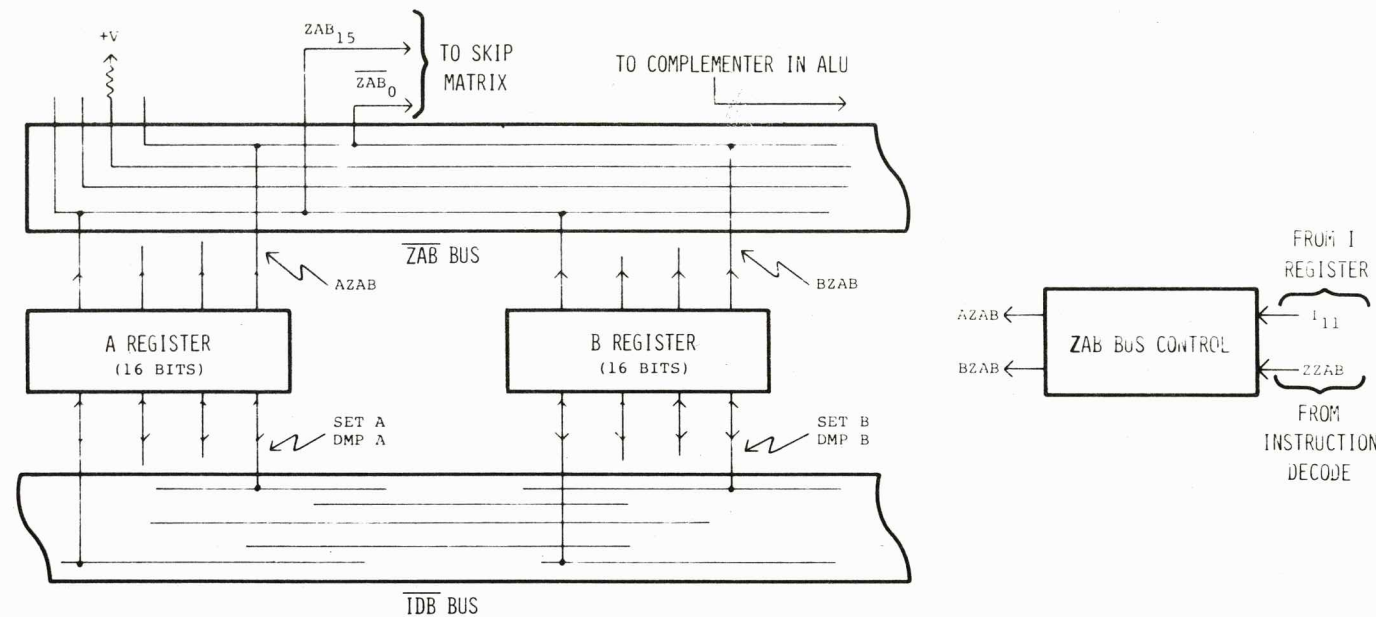


FIG 7-1

DETAILS OF THE A AND B REGISTERS

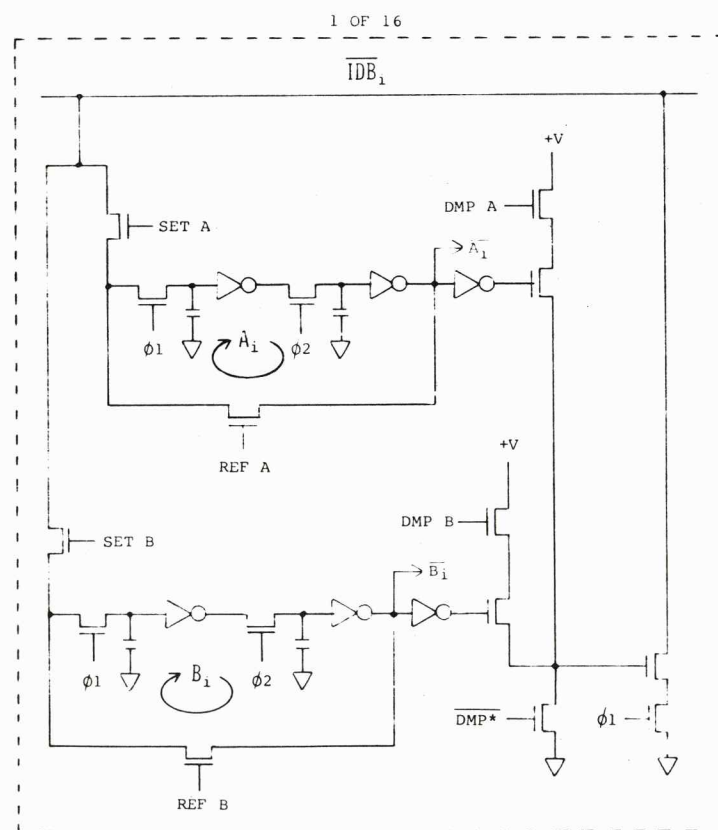
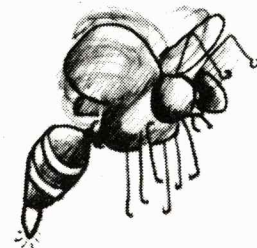


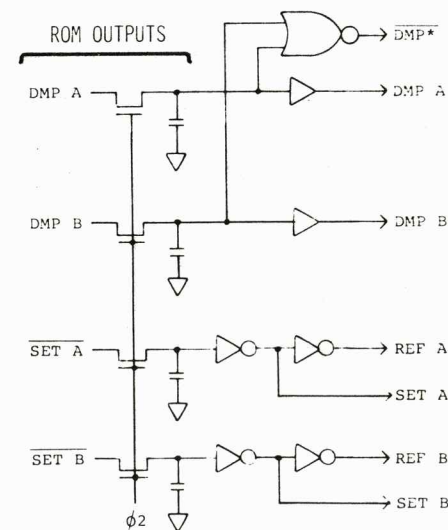
FIG 7-2

SECTION 7

Figure 7-1 is an overview of the A and B registers. Each of the A and B registers is connected to the IDB Bus via standard SET and DMP micro-instructions. In addition, each register contributes to the ZAB Bus. The main purpose of this Bus is to connect either the A or B register to the ALU. The ZAB Bus



THE "B" REGISTER



DETAILS OF THE ZAB BUS AND ZAB BUS CONTROL

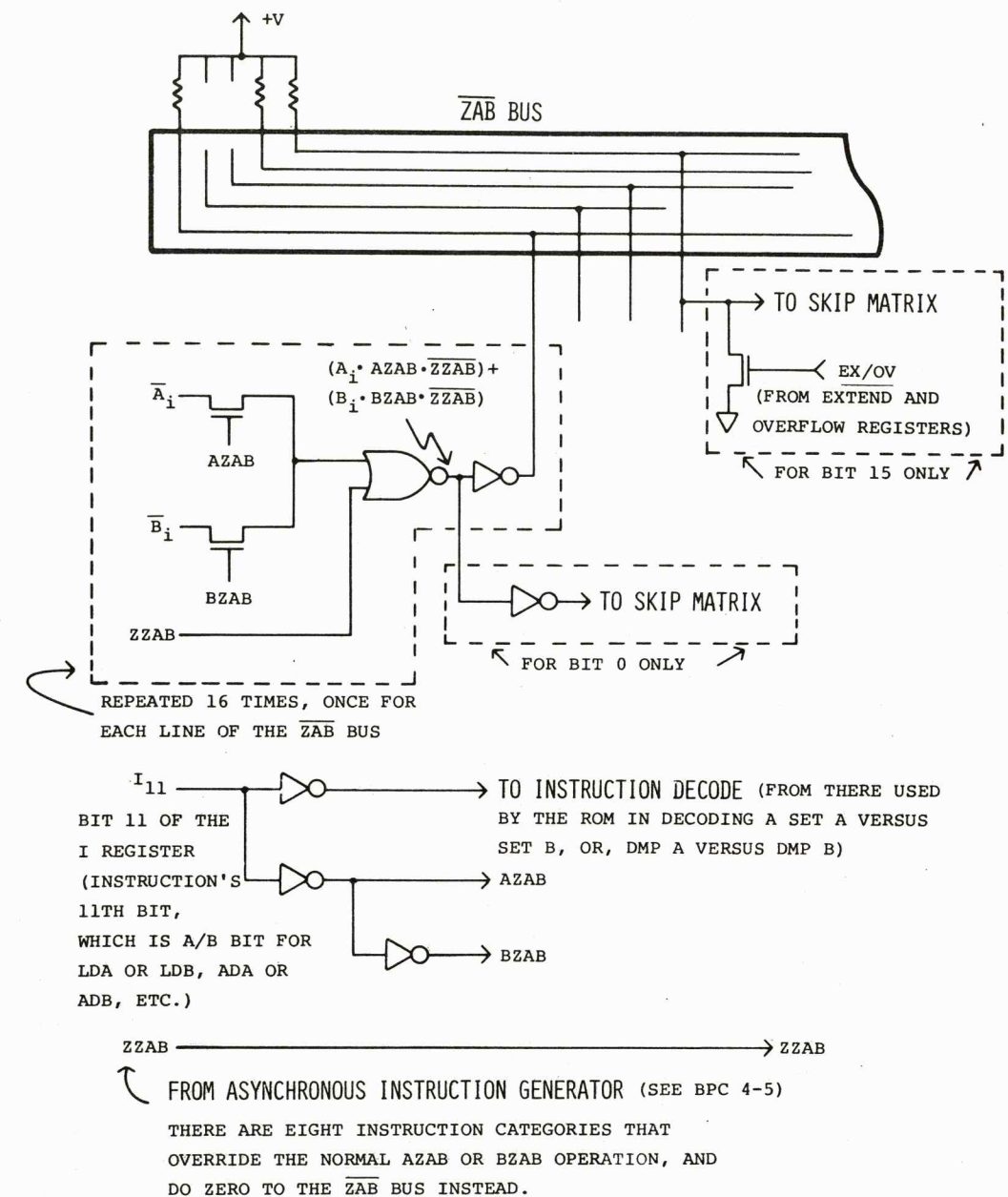


FIG 7-3

Control circuit determines whether the A register, B register, or neither, is connected to the ZAB Bus.

Figure 7-2 illustrates the actual circuitry of the register cells themselves. Observe the manner in which the A and B registers are tapped for use on the ZAB Bus. Notice also that the DMP A and DMP B micro-instructions share common pull downs on the IDB lines.

Figure 7-3 illustrates the details of the circuitry which controls the ZAB Bus. The ZAB Bus can represent either the A register, the B register, or all

zeroes. If the ZAB Bus is to represent all zeroes a signal called ZZAB is generated by Instruction Decode. ZZAB causes neither the A or B registers to be placed on the ZAB Bus. The ZAB Bus is then continually pulled-up by its pull-up resistors, and thus represents all zeroes. In the absence of ZZAB the ZAB Bus will represent either the A or B registers, as determined by bit 11 of the I register.

Observe that in the absence of ZZAB bit 0 of either the A or B register is picked off and sent to the Skip Matrix. This is used in the execution of the SZA, SZB, RZA and RZB machine-instructions.

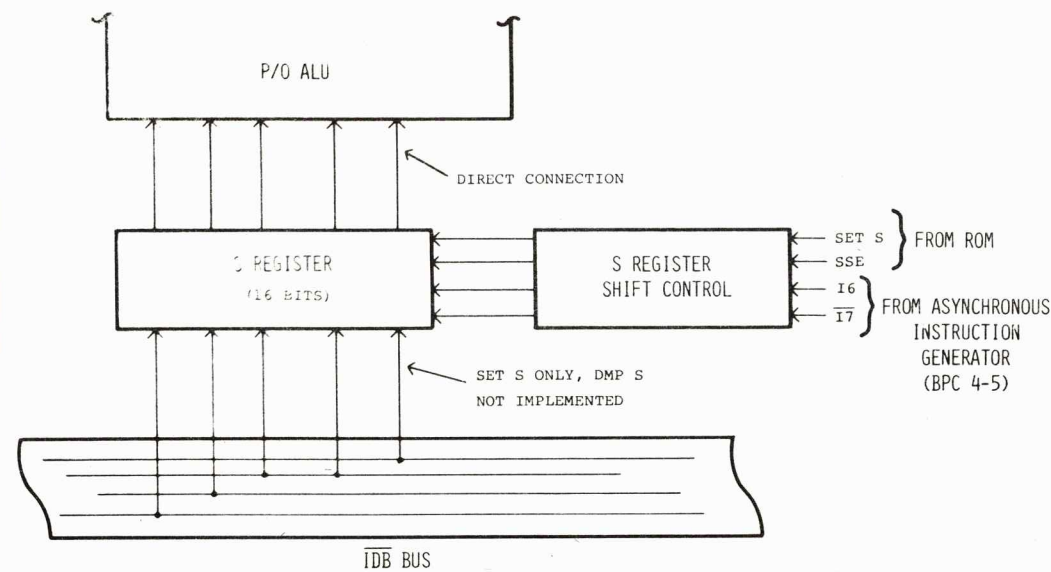
ZAB BUS CONTROL

A & B REGISTERS

R REGISTER

NSC ENCODER

OVERVIEW OF THE S REGISTER AND SHIFT CONTROL



S REGISTER OPERATIONS:
 SET S (IDB BUS TO S)
 RIGHT SHIFT WITH $0 \rightarrow S_{15}$
 ARITHMETIC RIGHT SHIFT (S_{15} UNALTERED)
 ROTATE S RIGHT ($S_0 \rightarrow S_{15}$)
 SHIFT S LEFT WITH $0 \rightarrow S_0$

FIG 8-1

DETAILS OF THE S REGISTER

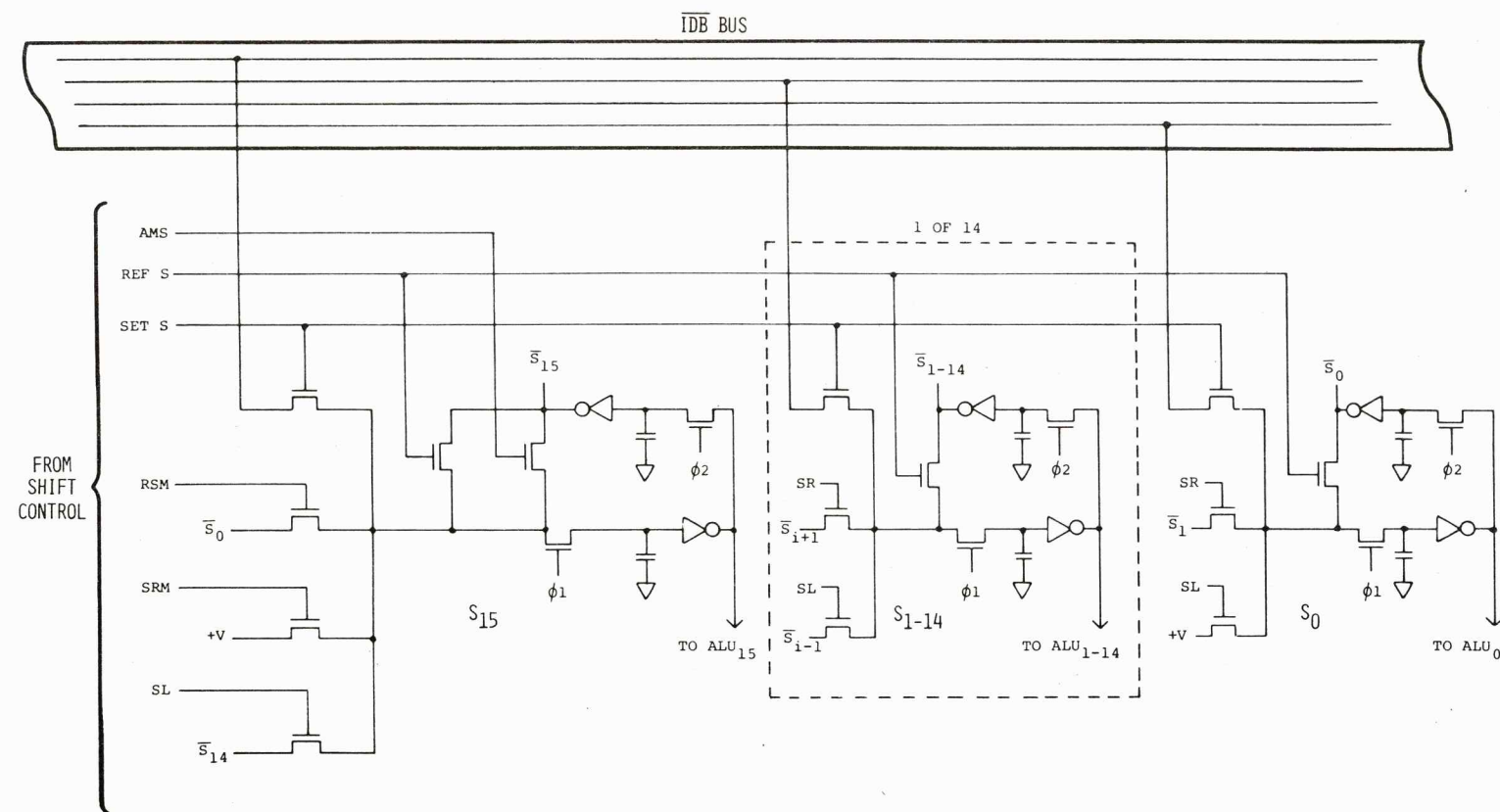


FIG 8-2

SECTION 7 (CONTINUED)

Also observe that the Extend and Overflow registers are capable of exerting the most significant ZAB line low. That line is also connected to the Skip Matrix, and is used in the execution of the SOS, SOC, SES, and SEC machine-instructions.

SECTION 8

Figure 8-1 is an overview of the S register and of its interconnections with the IDB Bus and the ALU. The S register is capable of a variety of shifting operations, and is used to perform the required

shifting in such machine-instructions as SAR and SAL. A Shift Control circuit uses two bits from the I register to determine what type of shift is to be performed when an SSE micro-instruction is given.

The S register has two functions. First, it is used to shift things. Second, it is used as an input to the ALU. Obviously then, there needs to be a SET S micro-instruction. There is, however, no DMP S. After S has been used to perform a shift operation, a DMP S is achieved by having the ALU add zero to the contents of the S register and by then doing a DMP ALU.

Figure 8-2 shows a typical S register cell and the manner in which the cells are interconnected together for shifting. The signal AMS is generated during the execution of the arithmetic shift machine-instructions AAR and ABR. AMS preserves bit 15 of the S register during right-shift operations. RSM is used to connect bit 0 to the input of bit 15 during a rotate-right operation. SRM is used to set bit 15 to a 0 during a shift-right operation. SL is used to shift left, and SR is used to shift right. Observe that each bit of the S register is sent via a direct connection to the ALU.

DETAILS OF SHIFT CONTROL

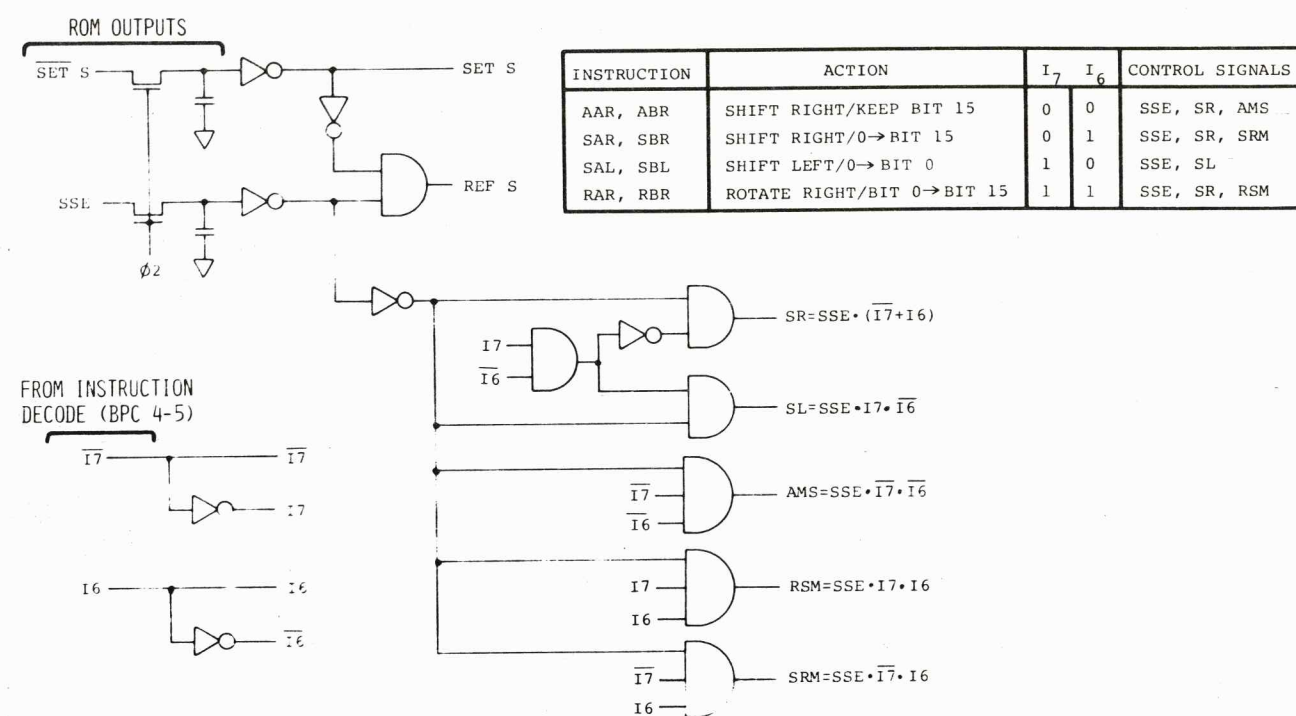


FIG 8-3

SECTION 8 (CONTINUED)

Figure 8-3 shows the details of the S register Shift Control circuitry. In particular, the table shows how various combinations of I register bits differentiate between the various shift machine-instructions, and shows which control signals arise as a result of those machine-instructions.

SECTION 9

Figure 9-1 is an overview of the ALU. The ALU is used to perform the arithmetic and logical operations implemented by the BPC. The ALU has two 16-bit data inputs; these are the ZAB Bus and the S register. The ZAB Bus can be complemented or not, according to the asynchronous control lines. The operations that the ALU can perform are: two's complement binary summation, logical summation (OR), and logical product (AND). Each of these operations is performed as a parallel operation on full 16-bit words.

The Adder portion of the ALU continually forms the AND, OR and SUM of the two input quantities. The control signals ANC, ORC and ADC

OVERVIEW OF THE ALU

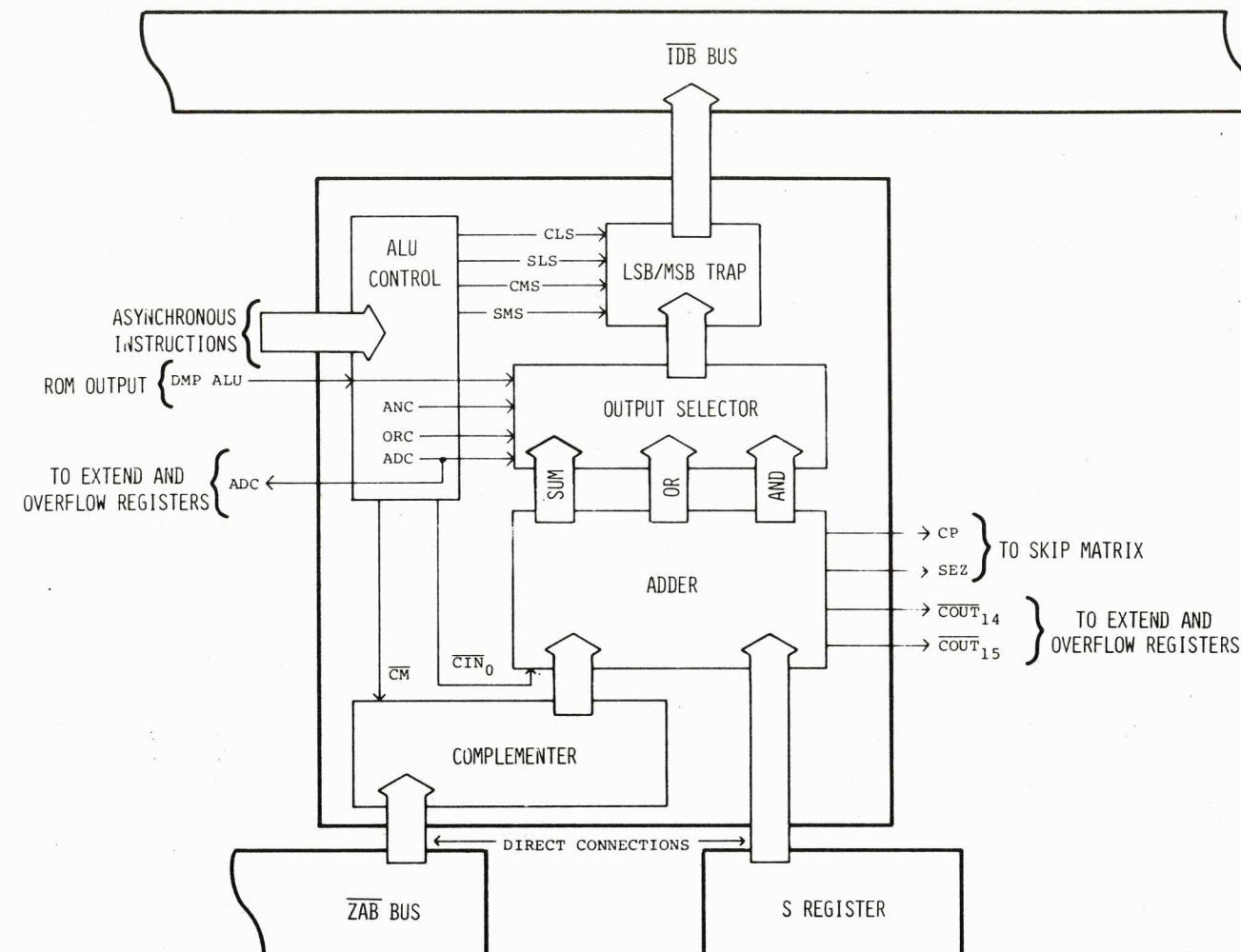


FIG 9-1

determine which of the outputs of the Adder will be made available to the IDB Bus during a DMP ALU. ANC makes the AND output available, ORC makes the OR output available and ADC makes the summation available.

The Complementer performs a straight one's complement. To do a two's complement operation the asynchronous control lines arrange that there be a carry-in on bit 0 of the Adder.

Certain outputs are available from the Adder on a continuous basis. CP indicates whether or not the two 16-bit quantities present at the inputs to the Adder itself are exactly identical. SEZ indicates whether or not the summation of those two quantities is exactly 0.

The two carry-outs are made available so that the Overflow and Extend registers may be properly set during a binary addition machine-instruction. It is for that purpose that ADC is also sent to the Extend and Overflow registers.

Certain Alter/Skip machine-instructions can set or clear either the most significant or least significant bit of the output of the ALU. The ability to do this is provided by the LSB/MSB Trap. It is in series with the output of the Adder and can force the desired bit to the desired state. It is also under the control the the asynchronous control lines.

OVERVIEW OF THE ADDER AND COMPLEMENTER

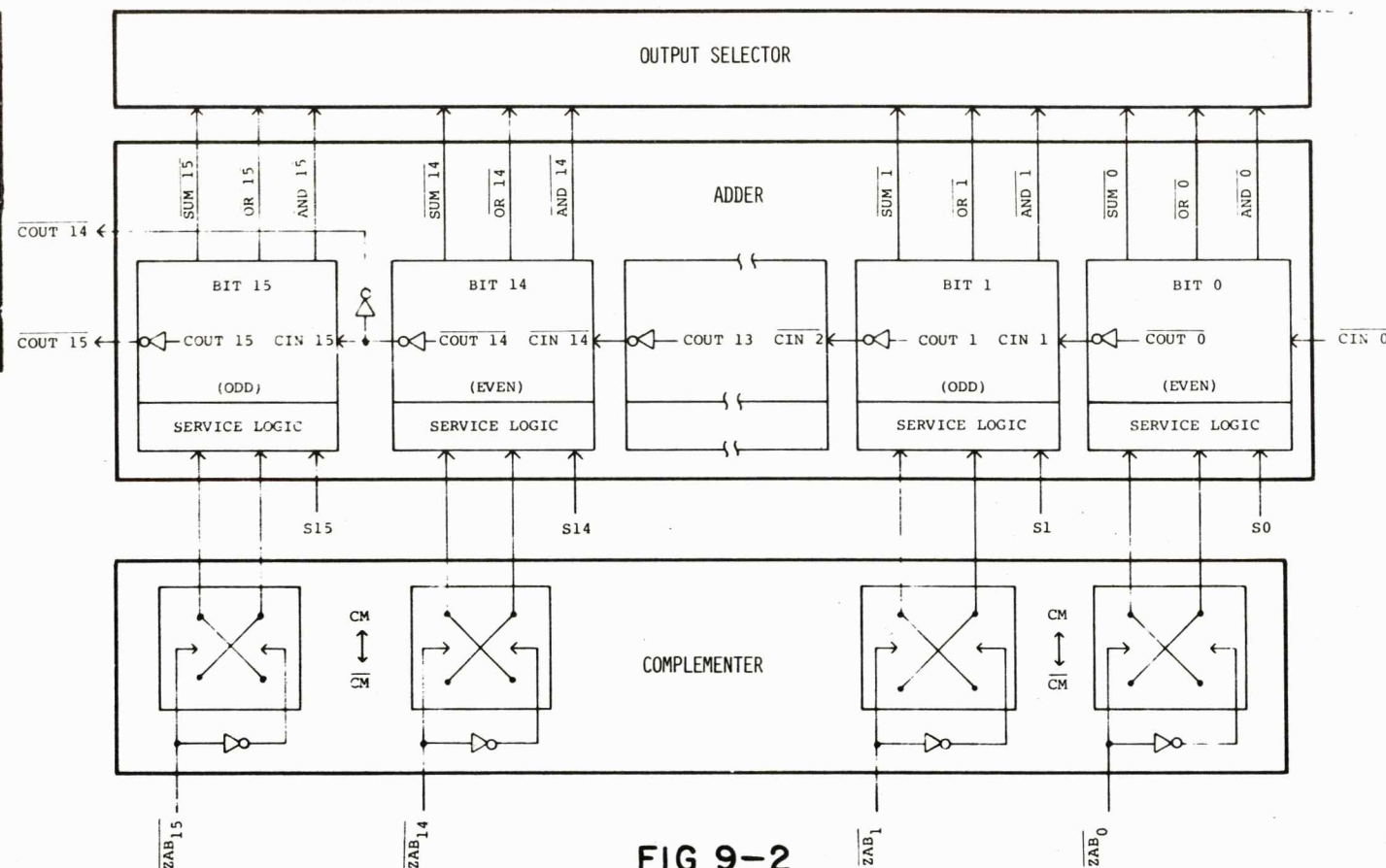


FIG 9-2

DETAILS OF THE COMPLEMENTER, SERVICE LOGIC, CP, AND SEZ CIRCUITS

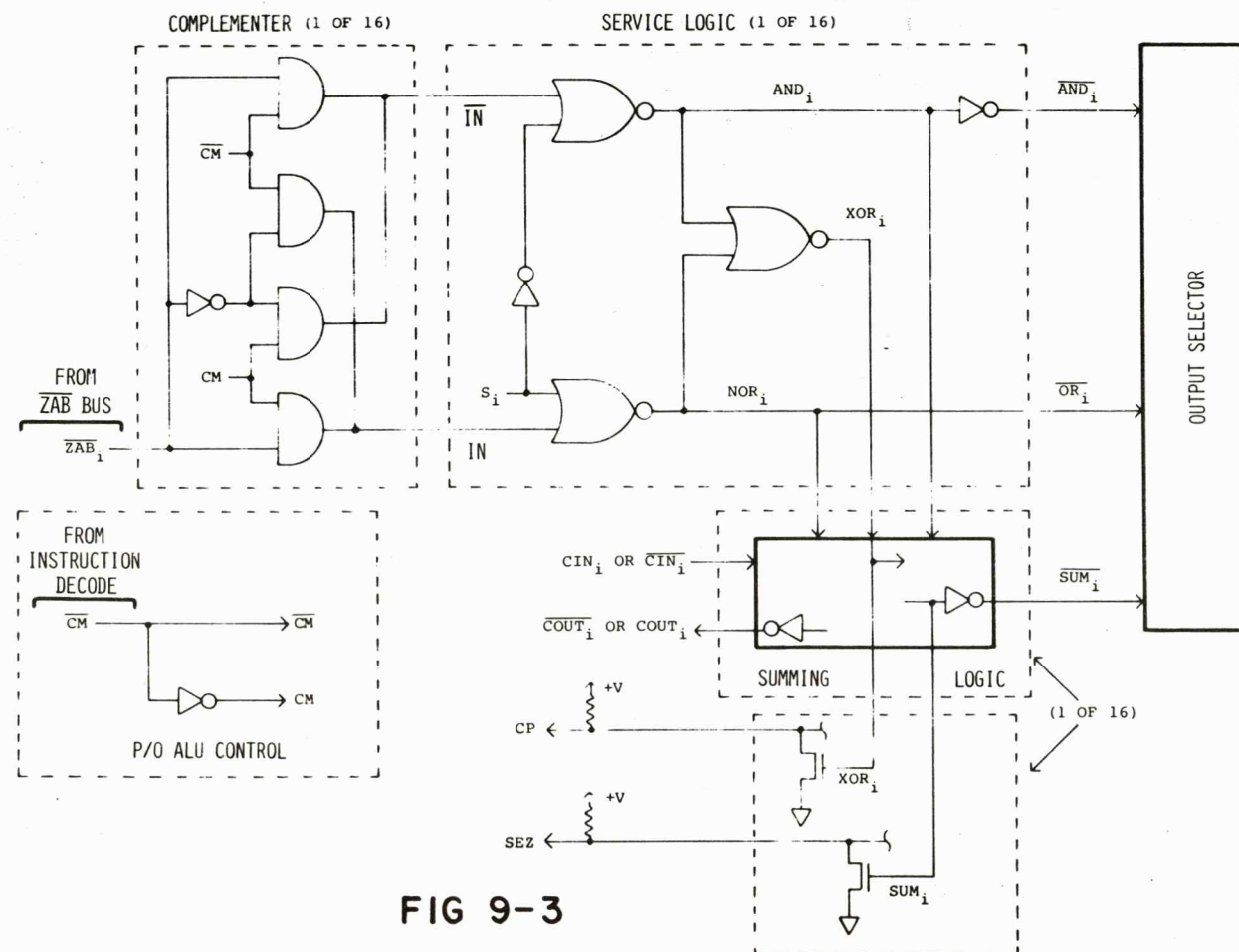


FIG 9-3

RULES FOR GENERATING SUM AND CARRY BITS

SERVICE LOGIC SIGNALS	INPUTS			OUTPUTS	
	CIN _i	ZAB _i	S _i	COUT _i	SUM _i
NOR	0	0	0	0	0
XOR	0	0	1	0	1
XOR	0	1	0	0	1
AND	0	1	1	1	0
CIN•NOR	1	0	0	0	1
CIN•XOR	1	0	1	1	0
CIN•XOR	1	1	0	1	0
CIN•AND	1	1	1	1	1

"AND", "NOR" AND "XOR" ARE MUTUALLY EXCLUSIVE CONDITIONS, AMONG WHICH EXACTLY ONE MUST ALWAYS BE TRUE. NOW:

- SUM_i=1 IF EXACTLY ONE OR THREE INPUTS (AMONG S_i, ZAB_i AND CIN_i)=1.
- COUT_i=1 IF AND_i+(XOR_i•CIN_i)=1.

A AND B CAN BE IMPLEMENTED BY CIRCUITRY THAT PERFORMS THE FOLLOWING OPERATIONS:

- IF NOR_i=1, THEN SUM_i=CIN_i; COUT_i=0 (FOR ODD BITS), COUT_i=1 (FOR EVEN BITS)
- IF XOR_i=1, THEN SUM_i=CIN_i; COUT_i=CIN_i
- IF AND_i=1, THEN SUM_i=CIN_i; COUT_i=1

FIG 9-4

SECTION 9 (CONTINUED)

Figure 9-2 is an overview of how the Complementer and the Adder function together. All the cells of the Complementer are identical. However, the cells of the Adder are of two types, depending on whether the cell is for an even or odd numbered bit. The change affects only the logical sense of the propagated carry, and was a device used to ensure that no single carry had to propagate too far without encountering a driver to revitalize it.

Figure 9-3 illustrates the details of the Complementer and of the Service Logic portion of the Adder. The Service Logic circuitry is identical regardless of whether it will be used for an even or odd numbered adder cell.

The Complementer circuitry is quite simple. It amounts to taking the input line, generating both logical senses of it, and then

connecting them to the Service Logic through a "reversing switch."

The Service Logic is more sophisticated. It takes the complemented or not ZAB input and the S input and then creates the AND, exclusive OR and NOR of each bit position. All three of these flavors are used by the even and odd summing logics to produce the sum. The AND and the NOR, however, can also be sent directly to the Output Selector.

One other use is made of the exclusive OR outputs; they are all NOR'ed together to produce CP. Notice also that the various sum bits are NOR'ed together to produce SEZ.

Figure 9-4 illustrates the rule used by the summation mechanism to generate the proper sum, given the NOR, exclusive OR and AND of the two input bits.

DETAILS OF SUMMATION AND CARRY PROPAGATION IN THE ALU

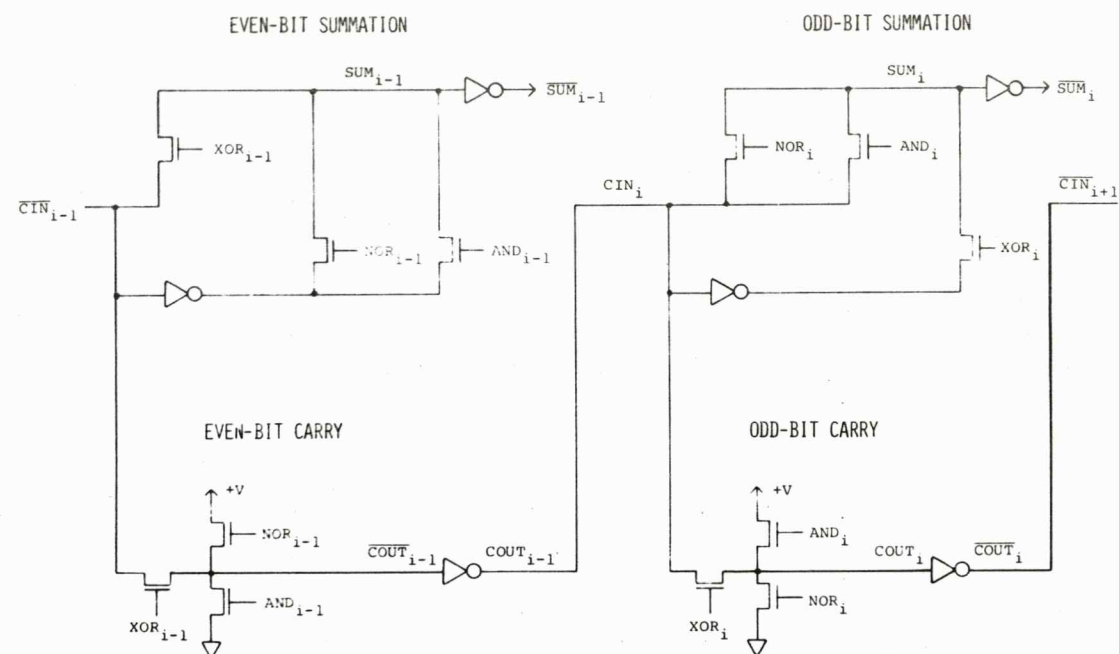


FIG 9-5

DETAILS OF THE OUTPUT SELECTOR AND LSB/MSB TRAP

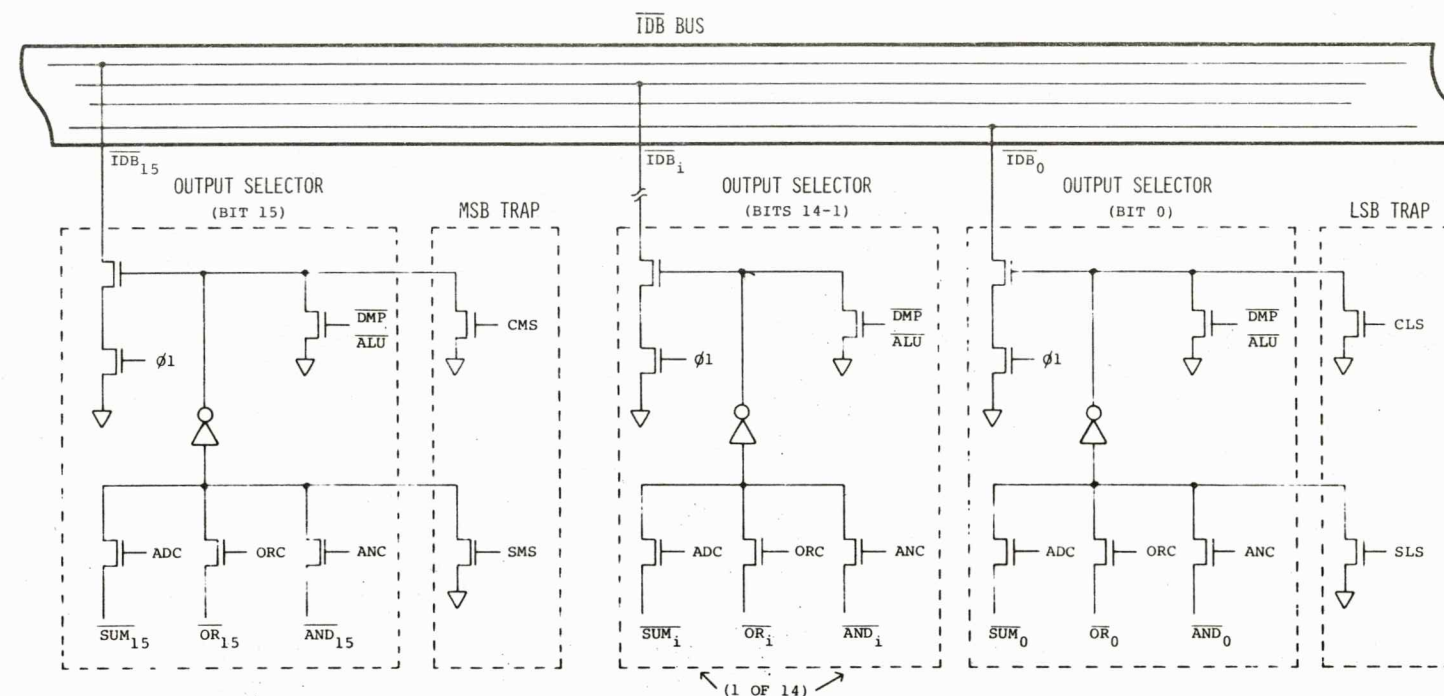


FIG 9-7

DETAILS OF ALU CONTROL

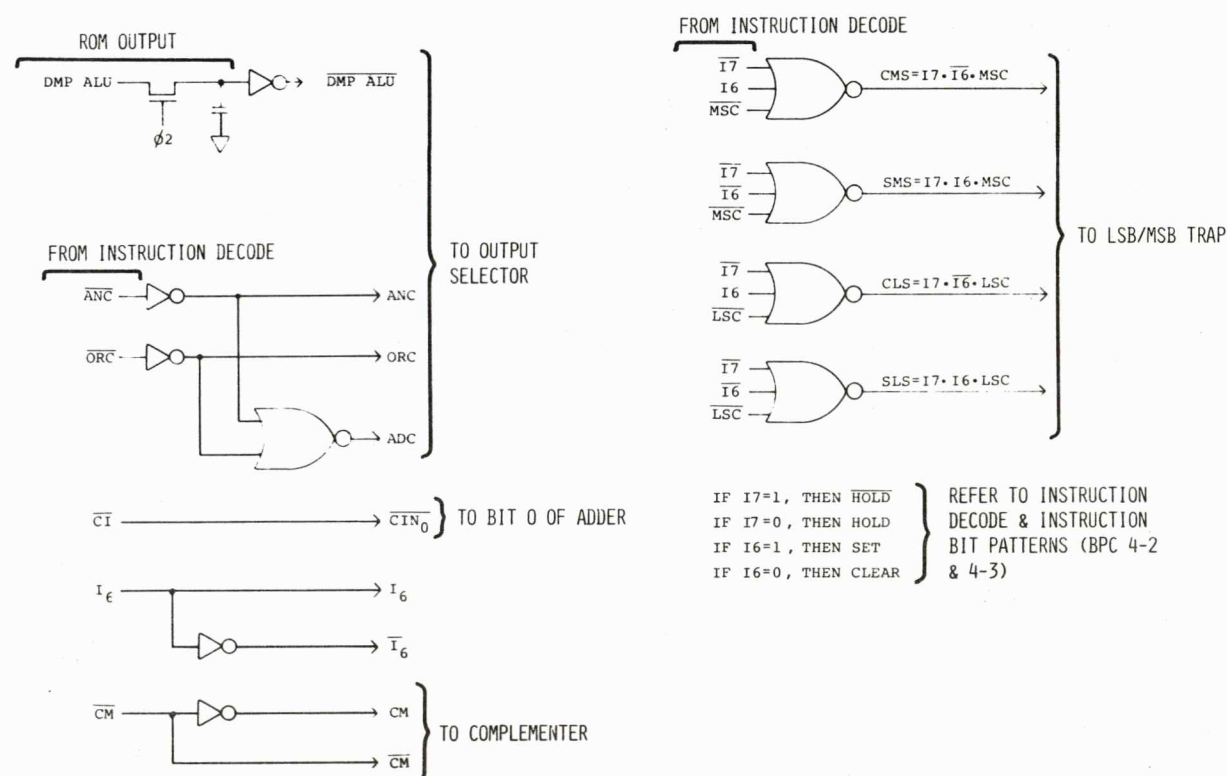
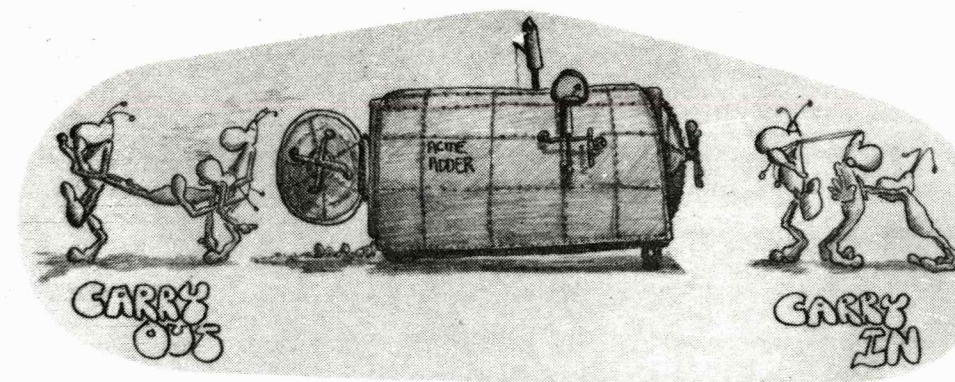


FIG 9-6



SECTION 9 (CONTINUED)

Figure 9-5 illustrates the actual summation mechanism. Observe how the logical sense of the carry bit changes at each bit position. The even cells require a different pattern of pull-ups and pull-downs to implement the table of Figure 9-4, than do the odd bit cells. However, each is just a straightforward implementation of the rules of Figure 9-4.

Figure 9-6 shows the details of the ALU Control circuitry. All that is being done here is to exploit the information available from Instruction Decode, concerning the nature of the ALU operation to be performed for the particular machine-instruction at hand.

Figure 9-7 shows the implementation of the Output Selector and the LSB/MSB Trap. The basic cell allows a DMP ALU to supply either the SUM, the OR or the AND of the two input quantities to the IDB Bus. The output selector for bit 15 has the MSB Trap appended to it. The control signal CMS forces the most significant bit to be a 0, while the control signal SMS forces it to be a 1. A like situation exists for the least significant bit. It has connected to it the LSB Trap, whose control signals CLS and SLS can clear or set the least significant bit, respectively.

ADDER & COMP'R OVERVIEW
SUM AND CARRY
BIT RULES
COMP'R, SER. LOGIC, CP AND SEZ
SUMMATION AND CARRY
ALU CONTROL
OUTPUT SELECTOR
AND LSB/MSB TRAP

OVERVIEW OF THE EXTEND AND OVERFLOW REGISTERS. ERA CAPABILITY NOT SHOWN

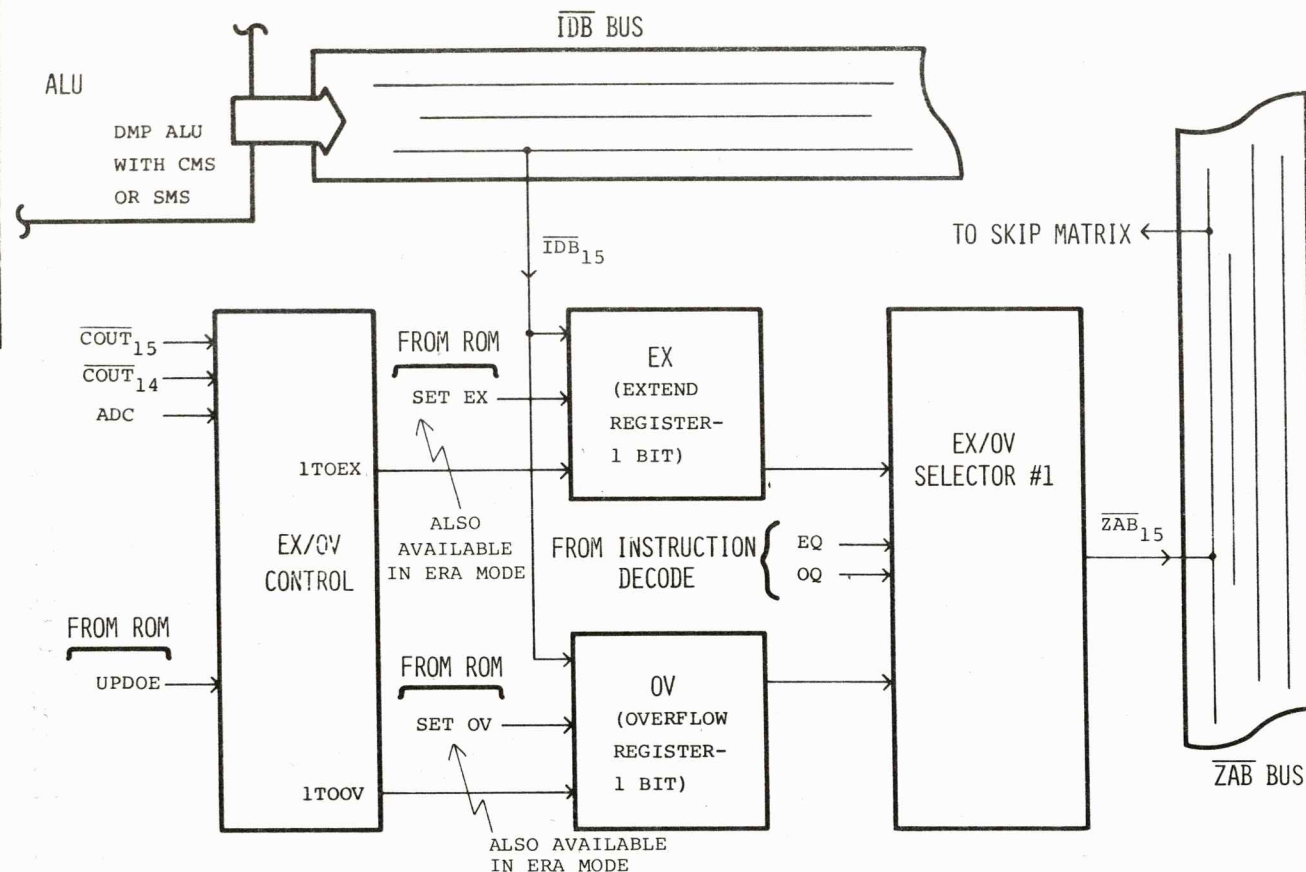


FIG 10-1

SECTION 10

Figure 10-1 is an overview of the normal usages of the Extend (E) and Overflow (OV) registers. It shows how they may be set or cleared as a result of two's complement arithmetic operations in the ALU, or as a result of the various Alter/Skip machine-instructions that deal with Extend and Overflow.

The Extend and Overflow registers are each 1-bit registers. The signal ITOEX causes an unconditional setting of the Extend register. The signal ITOOV causes an unconditional setting of the Overflow register. The EX/OV Control circuitry generates those two signals according to the carries from the Adder and to a ROM micro-instruction called Update Overflow and Extend (UPDOE). ITOEX is generated whenever UPDOE is given while the ALU is doing an ADD and there is a carry-out from bit 15. ITOOV is generated whenever there is a UPDOE and the exclusive OR

of the two carry-outs is true. The mechanism just described pertains to arithmetic only, and can set but not clear each of the Extend and Overflow registers.

The machine-instructions SES, SEC, SOS and SOC require additional mechanisms for their execution. Under certain circumstances bit 15 of the IDB Bus is used to represent either the Extend or the Overflow bit. During an alter-type machine-instruction the ALU can control the setting of that bit with the MSB Trap. Recall that the MSB Trap can force either a 1 or a 0. The SET EX and SET OV micro-instructions connect bit 15 of the IDB Bus to the Extend and Overflow registers, respectively. In this way, the Extend and Overflow registers can be set or cleared in response to the execution of machine-instructions.

A means is required to interrogate the value of the Extend

THE EXTEND AND OVERFLOW REGISTERS WITH ERA

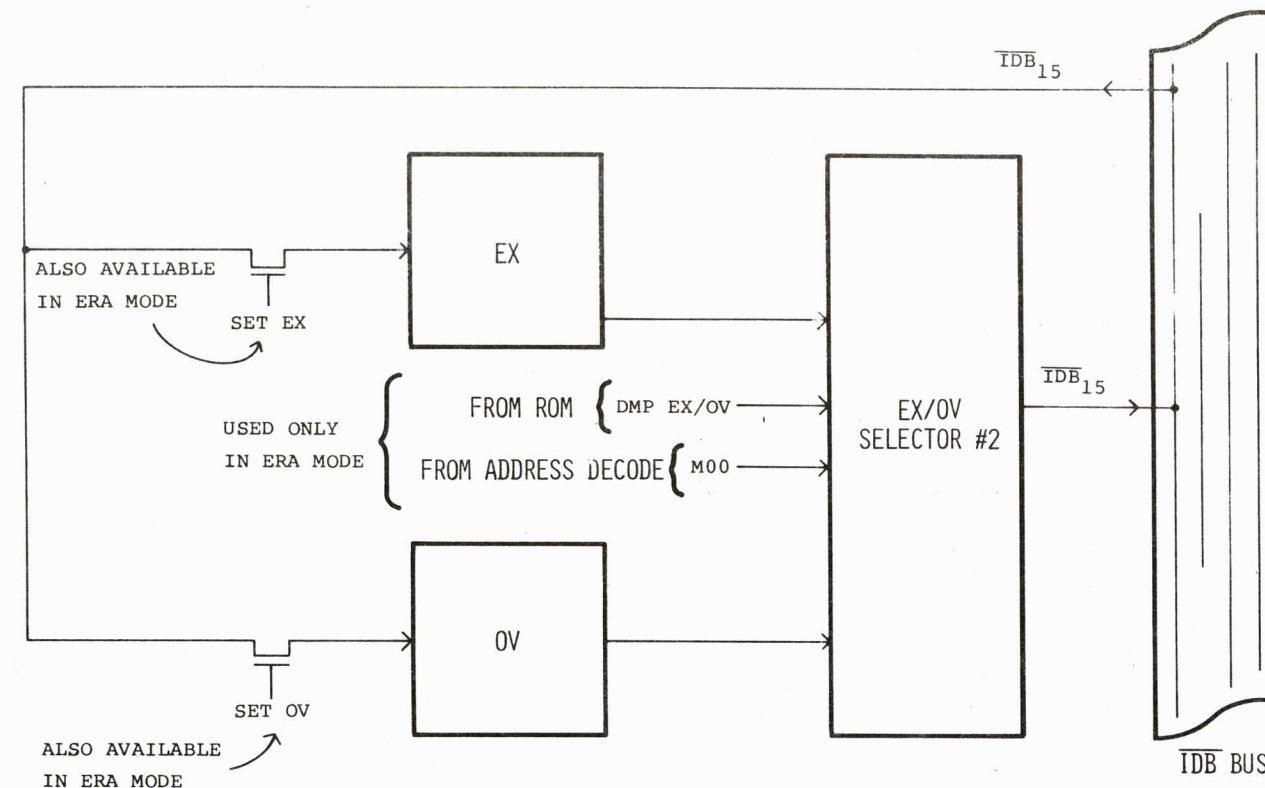


FIG 10-2

and Overflow registers. The circuitry of the EX/OV Selector #1 provide this means. In response to EQ and OQ from Instruction Decode, it selects the desired register output and utilizes ZAB(15) to connect the output to the Skip Matrix. In this way the "Skip" machine-instructions concerning Extend and Overflow can interrogate to those registers.

Figure 10-2 illustrates a second use of the Extend and Overflow registers. This is their use in the extended register addressing (ERA) mode of operation. In this mode the Extend and Overflow registers are given register addresses, just as have A and B. A special line (ERA) indicates that the special addressing mode is in effect.

During ERA mode addressing, the device that recognizes memory addresses corresponding to entities within the BPC will recognize the

appropriate addresses for Extend and Overflow, and generate the necessary SET and DMP micro-instructions to transfer the data. IDB(15) is the line used to transfer the data.

Special circuitry is used to allow the interrogation of the Extend and Overflow registers during the ERA mode of operation. The EX/OV Selector #2 uses a combined DMP EX/OV signal in conjunction with an address bit to select the register to be dumped onto IDB(15).

DETAILS OF EX/OV CONTROL

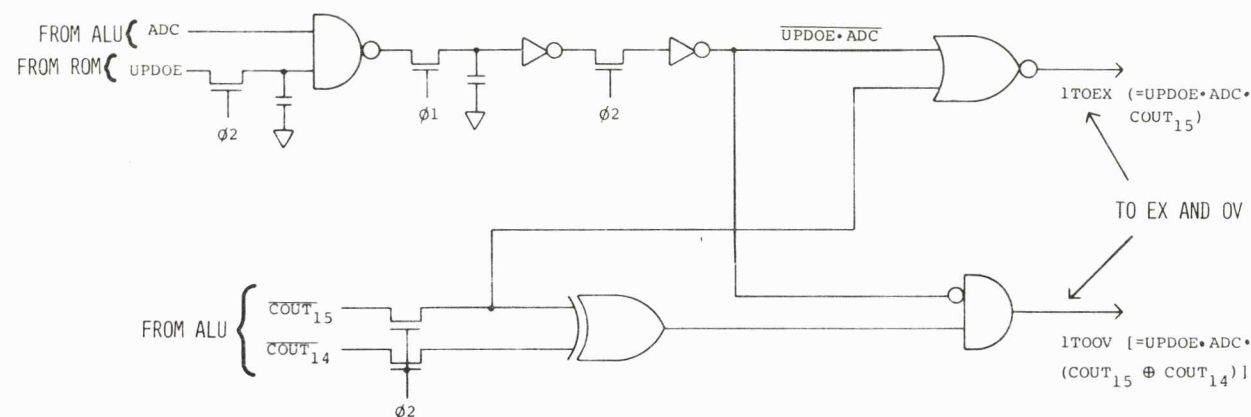


FIG 10-3

DETAILS OF EXTEND, OVERFLOW, SET EX, SET OV, AND EX/OV SELECTOR #1. ERA CAPABILITY NOT SHOWN

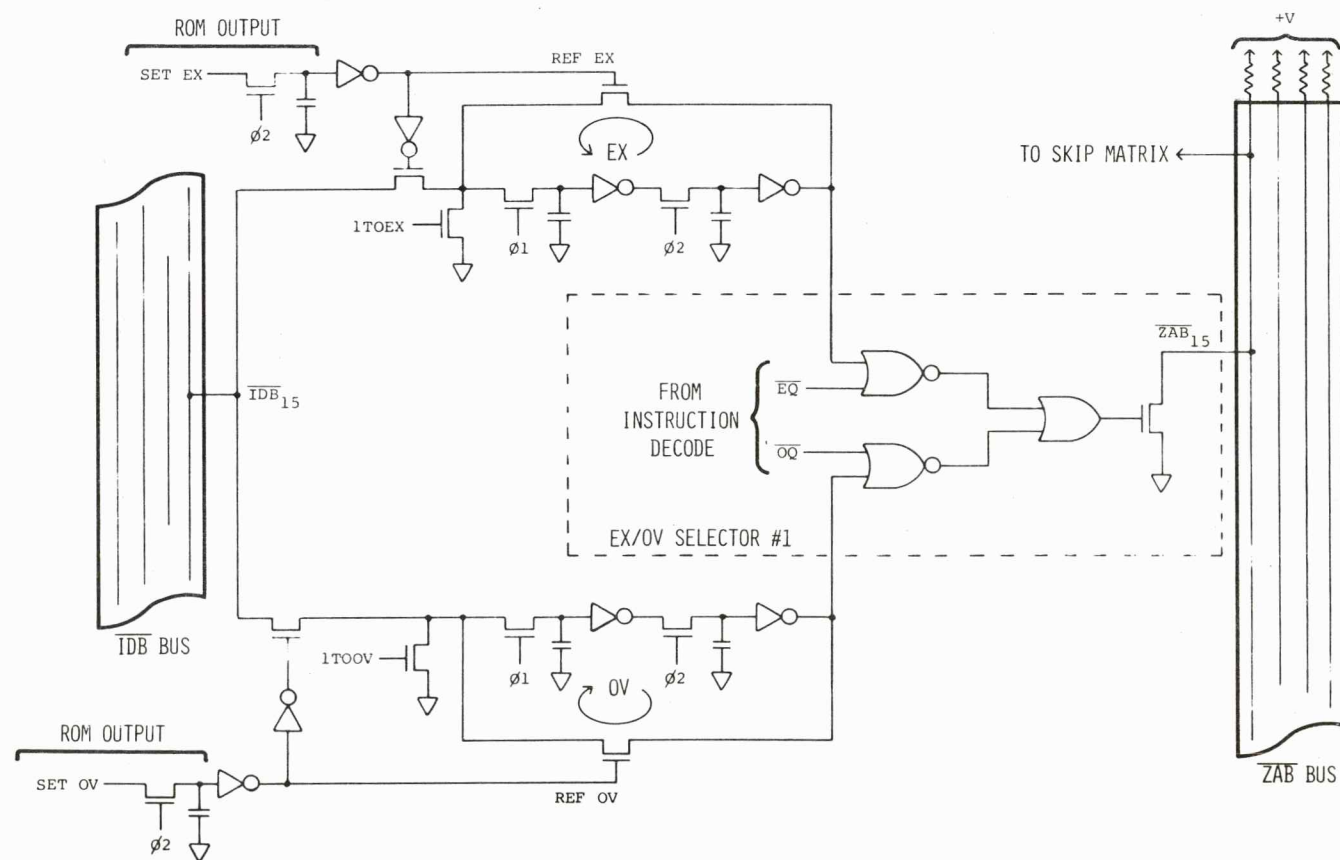


FIG 10-4

EXTEND AND OVERFLOW WITH EX/OV SELECTOR #2. SHOWS EX/OV/IDB BUS CAPABILITY IN ERA MODE

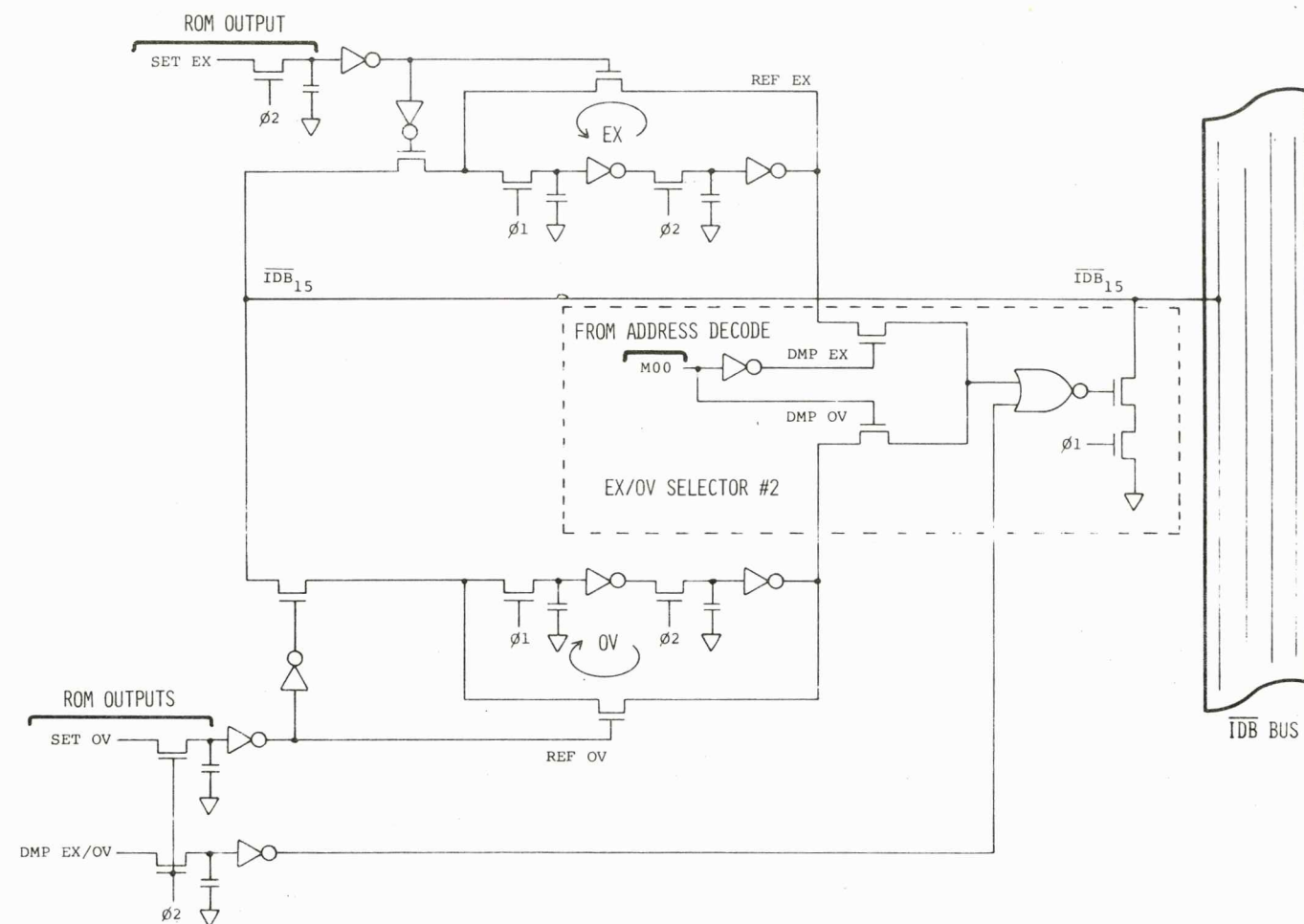


FIG 10-5

SECTION 10 (CONTINUED)

Figure 10-3 illustrates the details of the circuitry of EX/OV Control. Its purpose is solely to generate the signals ITOEX and ITOOV.

Figure 10-4 illustrates the circuitry of the Extend and the Overflow registers as well as EX/OV Selector #1. This is the circuitry used in non-ERA mode operation.

Figure 10-5 shows the details of Extend and Overflow operation when used in the ERA mode of operation with EX/OV Selector #2.

EX AND OV WITH EX/OV SELECTOR #2
EX/OV CONTROL EX, OV, SET EX, SET OV AND EX/OV SELECTOR #1
EX AND OV WITH ERA
NON-ERA OVERVIEW OF EX AND OV

THE FLAG MULTIPLEXER

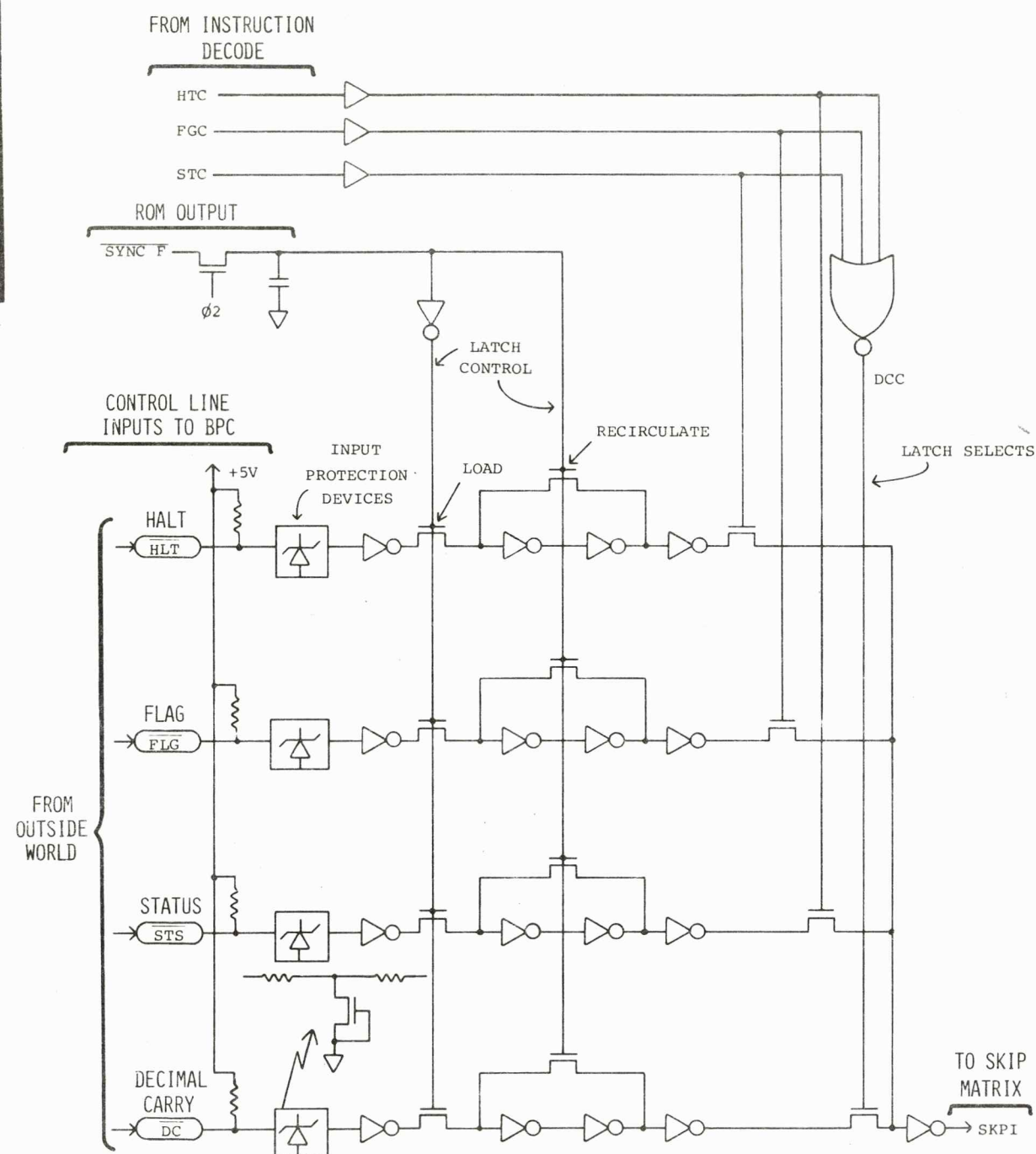
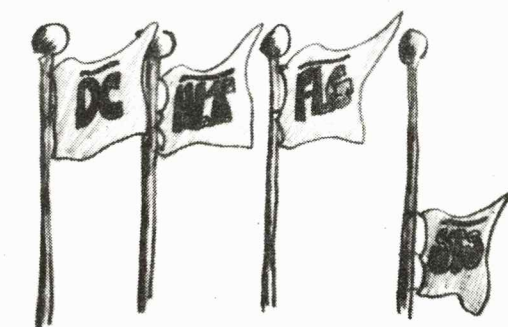


FIG 11-1



PROCESSOR FLAGS

THE SKIP MATRIX

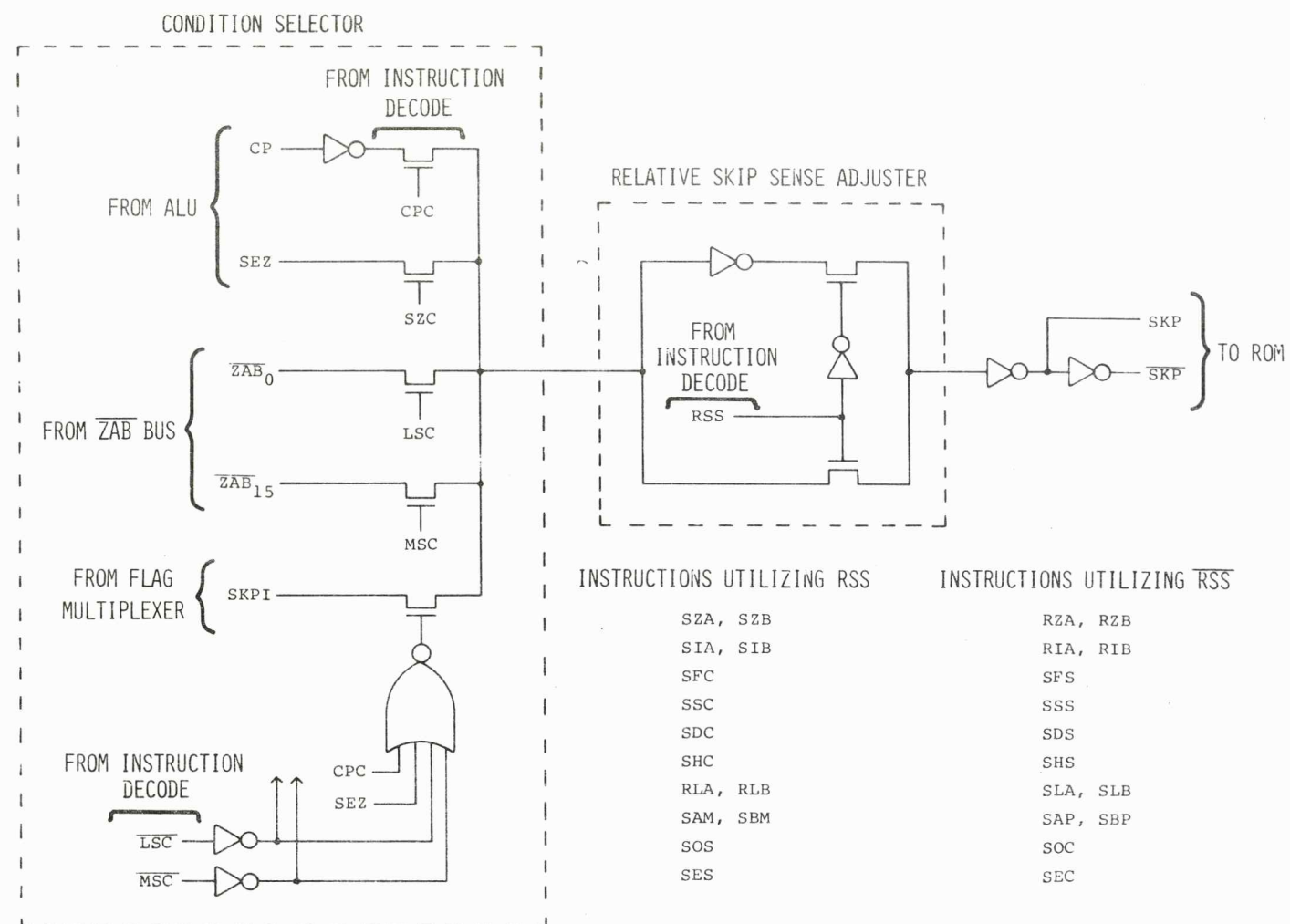


FIG 12-1

SECTION 11

Figure 11-1 illustrates the circuitry of the Flag Multiplexer. It has two functions. First, upon command of the ROM it latches in the values of the four external flags, HALT, FLAG, STATUS and DECIMAL CARRY. Second, it selects one of these values according to the asynchronous control lines and supplies that value to the Skip Matrix.

The micro-instruction SYNC F is the command to latch the flags. It is given by the instruction fetch portion of the BPC's ASM chart. At the conclusion of the memory cycle which fetches an instruction the values of the flags are latched. Shortly thereafter the instruction bit pattern is in the I Register. Instruction Decode will then generate the necessary control signals to select which latch output is to be

sent to the Skip Matrix. One of the latches is always transmitted, even if it will not be used.

SECTION 12

Figure 12-1 depicts the circuitry of the Skip Matrix. The purpose of the Skip Matrix is to generate and send to the ROM a qualifier called SKP. The flow charting encoded in the ROM will use that qualifier to determine whether or not to skip when executing a skip-type machine-instruction.

The Skip Matrix selects the condition to be tested, adjusts its logical sense according to the existence of a reverse-skip condition, and generates SKP. Instruction Decode determines which condition is selected and the relative logical sense of the skip operation.

THE P REGISTER

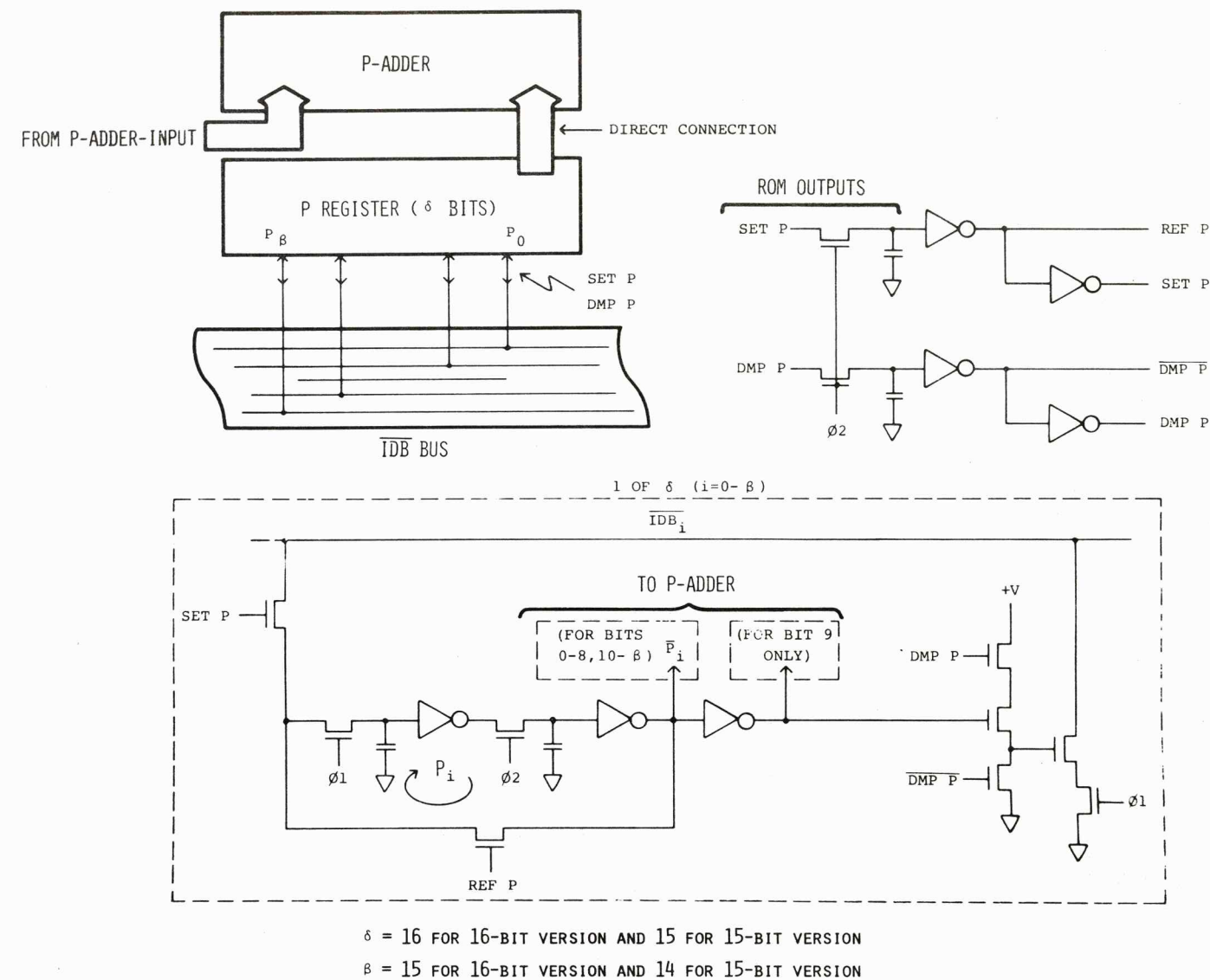


FIG 13-1

SECTION 13

Figure 13-1 depicts the P register. The P register is connected to the IDB Bus through both a SET P and a DMP P. Observe that the P register is connected to the P-Adder by a direct connection. Notice also that of the bits sent to the P-Adder, bit 9 is of the opposite sense from all the others.

The only difference between the 15 and 16-bit versions is that the P register in the 16-bit version is a full 16-bits, and that the extra bit is also sent to the P-Adder.

THE T REGISTER

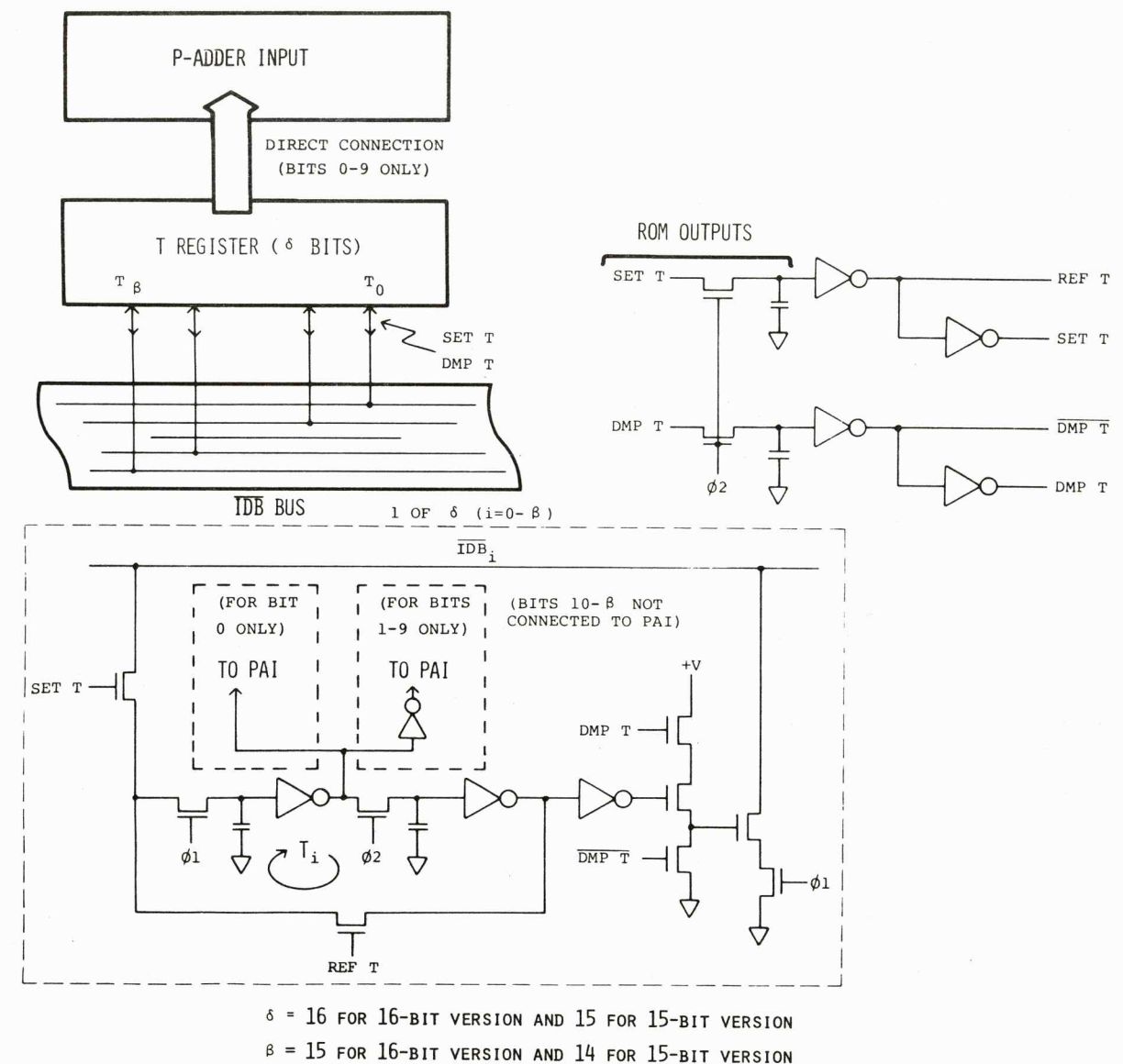


FIG 14-1

SECTION 14

Figure 14-1 depicts the T register. The T register is connected to the IDB Bus through both a SET T and a DMP T. The least 10 significant bits of the T register are sent by direct connection to the P-Adder Input (PAI).

The only difference between the 15 and 16-bit versions is that in the 16-bit version the T register is a full 16-bits; the extra bit is *not* sent to the PAI.

OVERVIEW OF THE PROGRAM ADDER MECHANISM

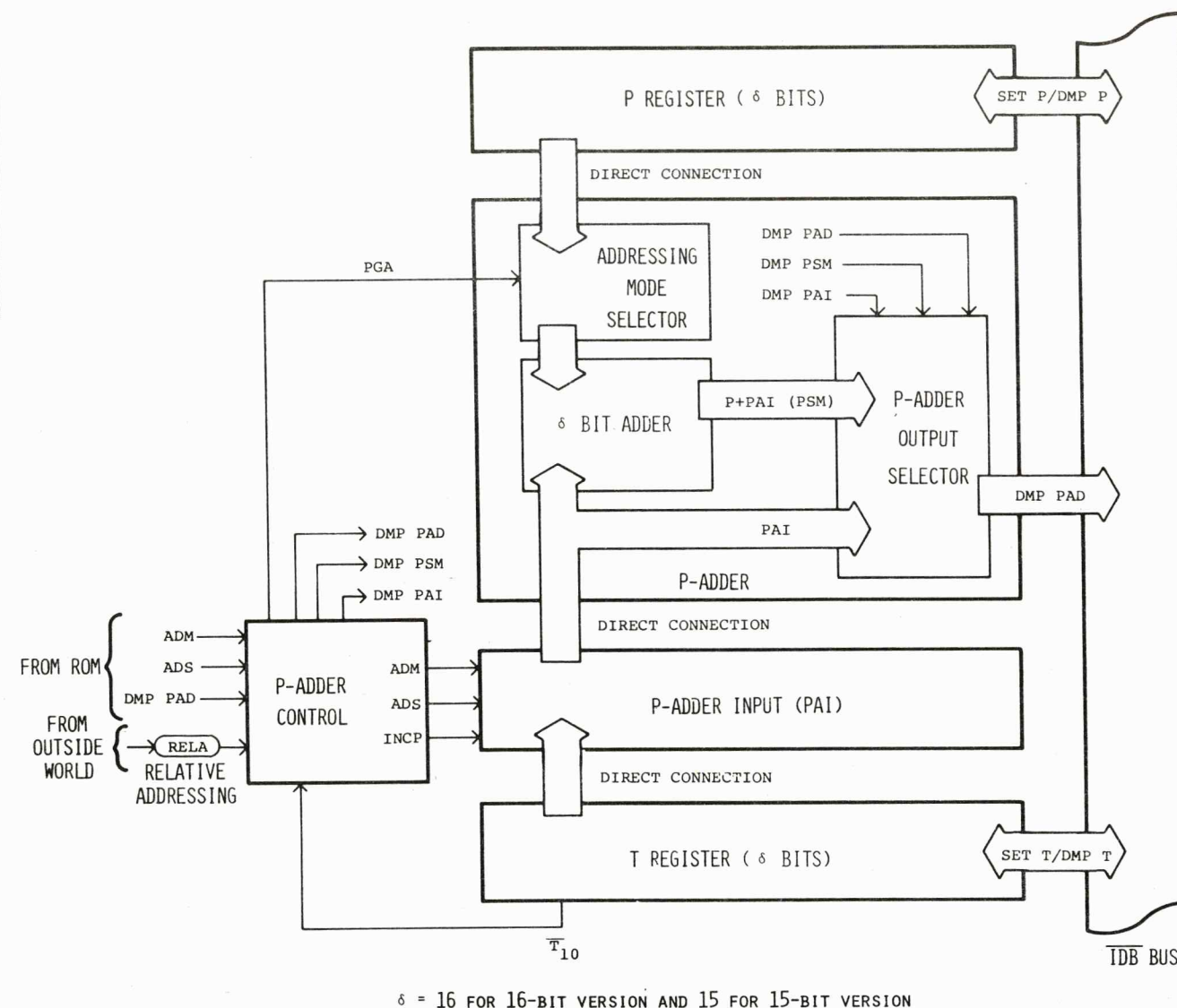


FIG 15-1

SECTION 15

Figure 15-1 is an overview of the program adder mechanism. The function of the P-Adder and its associated circuitry is to perform selected types of arithmetic upon the contents of the P register. There are three general cases that require the contents of P to be taken as the basis for an arithmetic operation. The first of these are the instruction fetches required in sequential program execution. To perform such fetches successive values of P are required; each value is one greater than the previous value. In general, some means must be employed to increment the value of P after each instruction fetch. The second case concerns

the formulation of addresses associated with memory reference machine-instructions. The ten least significant bits of the instruction bit-pattern of a memory reference instruction contain addressing information which must be combined with the contents of the P register to produce the final address. The third general category of P register arithmetic concerns the formation of new values for the P register based on the outcome of any of the various types of "Skip" machine-instructions. The P-Adder operates in conjunction with a P-Adder Input circuit and a P-Adder Control circuit.

The P-Adder Control circuit determines which kind of operation the P-Adder will perform. Because it is the most often needed result, and because it takes some time to perform it, the P-Adder will normally automatically form the value of P+1. It will do this on a steady-state basis in the absence of instructions to the contrary. The output of the P-Adder is available through the DMP PAD.

Unlike most other control circuits, the P-Adder Control circuit does *not* respond to Instruction Decode. Instead, it responds directly to micro-instruction decoded from the ROM. As mentioned above, in the absence of any control instructions P-Adder Control will cause the formation of P+1. It does this by generating INCP. INCP causes the P-Adder Input to totally ignore the contents of the T register and generate as its output a value of 1. The P-Adder adds the 1 and the value of P in the Adder itself, and the sum is made available to the P-Adder Output Selector. Because this is an operation requiring the actual output of the Adder, P-Adder Control will also generate DMP PSM. Then when DMP PAD is given the incremented value of P is placed on the IDB Bus.

The two micro-instructions that can alter the operation of the P-Adder are ADM and ADS. Neither is latched and both must be repeatedly given for the entire duration of the type of operation that they cause. The ADM micro-instruction causes the formation of memory addresses corresponding to memory reference machine-instructions. It does this by combining the value of P with the 10 operand bits placed in the machine-instruction by the assembler. At the time the memory reference machine-instruction was fetched it was placed in the T register. If absolute addressing is in effect, as determined by REL A, the Addressing Mode Selector suppresses the nine least significant bits of the P register and induces a half page offset by setting the tenth bit to a one. The modified P register contents are then added

to the bottom ten bits of the T register. However, the ADM causes the P-Adder Input circuit to propagate the tenth bit of T leftward prior to the addition. The tenth bit was set by the assembler to undo the half page offset introduced by the Addressing Mode Selector. As before, a DMP PAD in conjunction with a DMP PSM produces the desired output.

Relative addressing is similar, except that the bottom bits of the P register are not suppressed, and are taken for what they are.

Base page addressing for memory reference instructions operates as follows. T10 will be true, indicating that the memory reference instruction references the base page. That, in conjunction with ADM, causes DMP PSM to be false and DMP PAI to be true. In addition, the P-Adder Input circuit shifts the most significant bit of the address field in T all the way to the left. This creates the proper base page address in all cases, whether for relative or absolute, or for upper or lower half of the base page. All that is needed now is to do a DMP PAD while DMP PAI is true, and the base page address will appear on the IDB Bus.

The remaining mode of P-Adder operation concerns the formation of new values of P based on the outcome of a skip instruction. Such operation is performed under the auspices of the ADS micro-instruction. ADS causes the P-Adder Input circuit to propagate the left-most bit of the 6-bit skip field in the T register to the far left. This generates a two's complement number that can be added to the exact contents of the P register. After a suitable delay a DMP PAD and a DMP PSM will generate the desired new value of P, and place it onto the IDB Bus.

The only difference between the 15 and 16-bit versions is that in the 16-bit version the P register, T register and Adder are all 16-bit entities. This also causes a change in the P-Adder Input circuit, as there is now an extra bit to be propagated to the left.

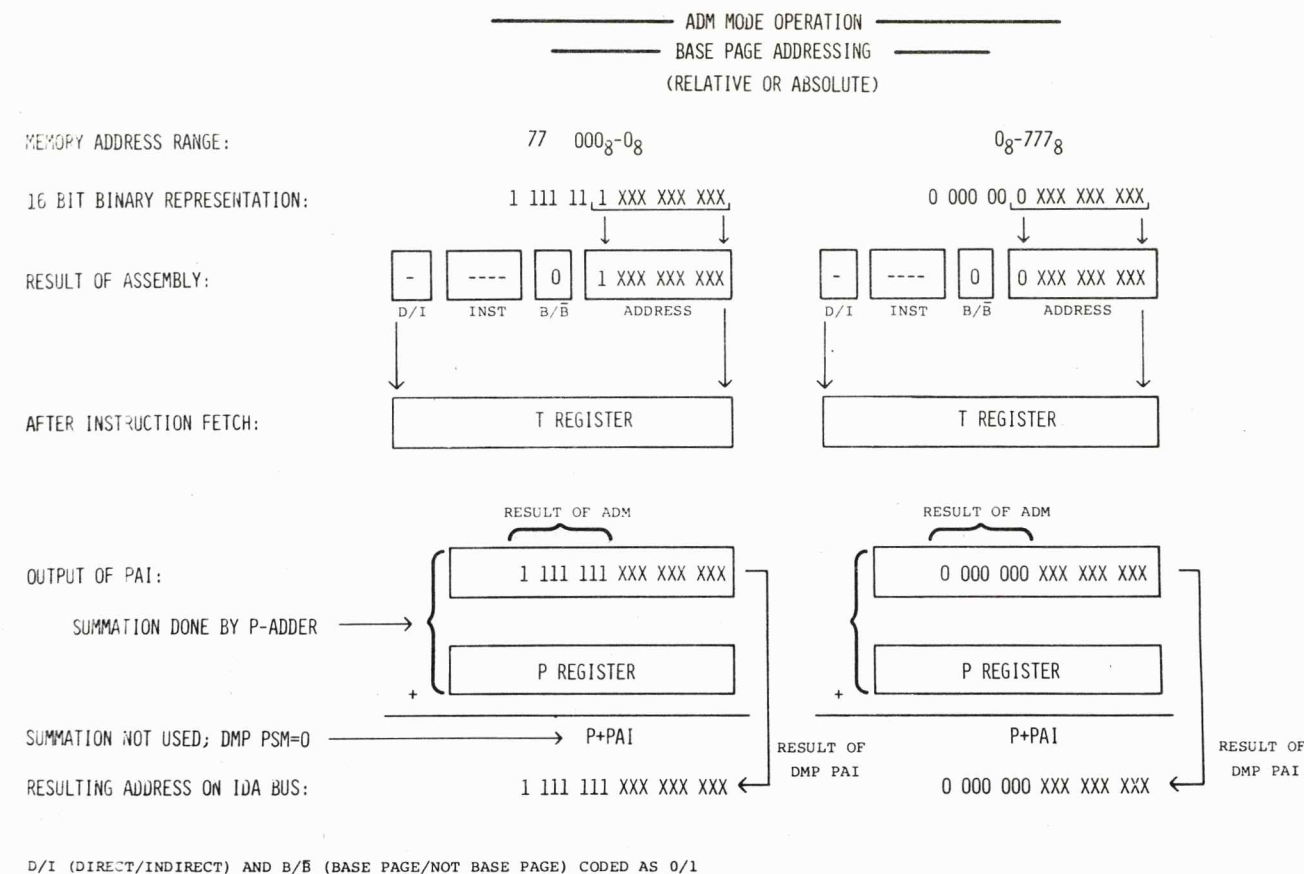


FIG 15-2-1-S

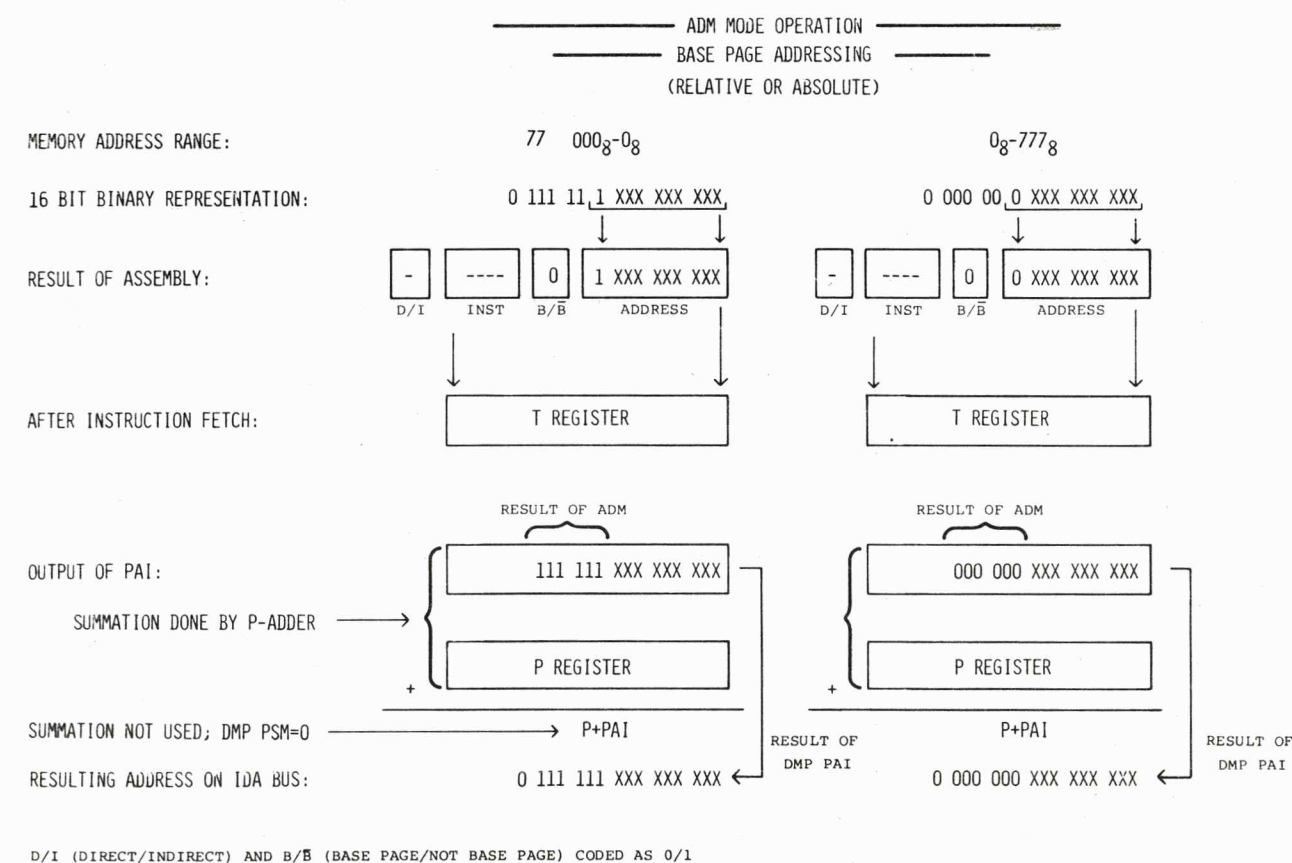


FIG 15-2-1-F

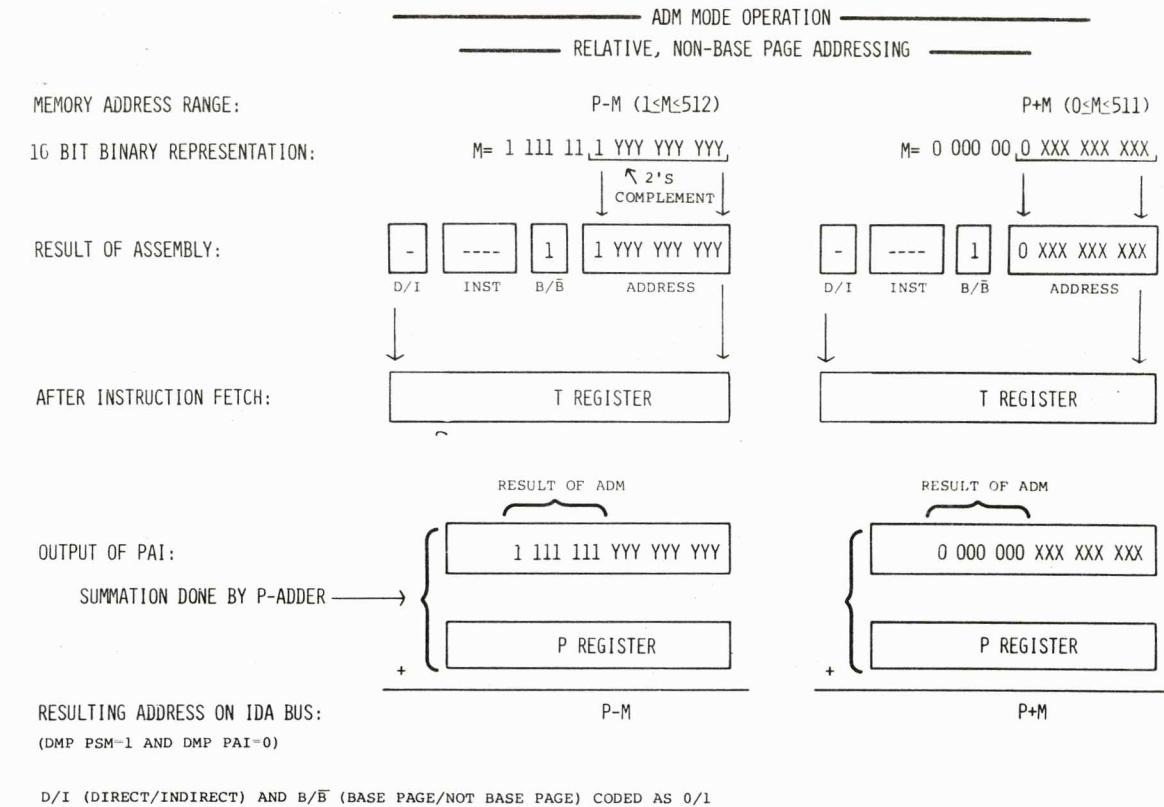


FIG 15-2-2-S

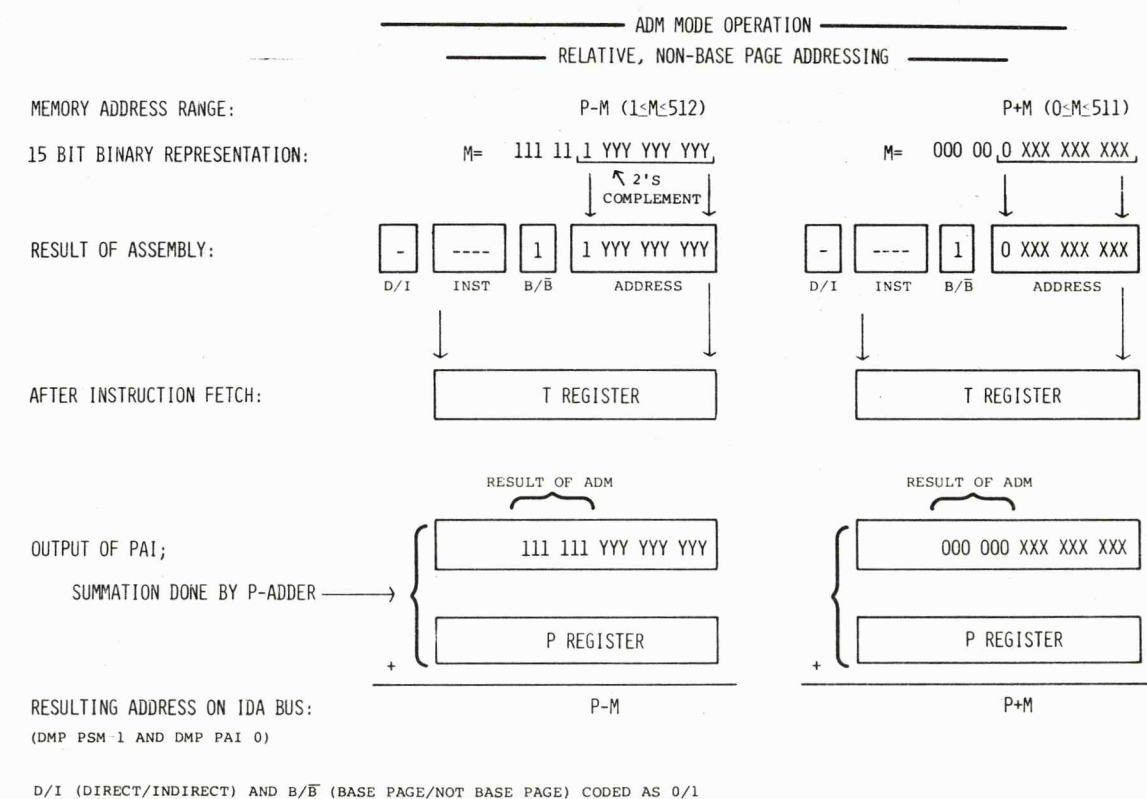
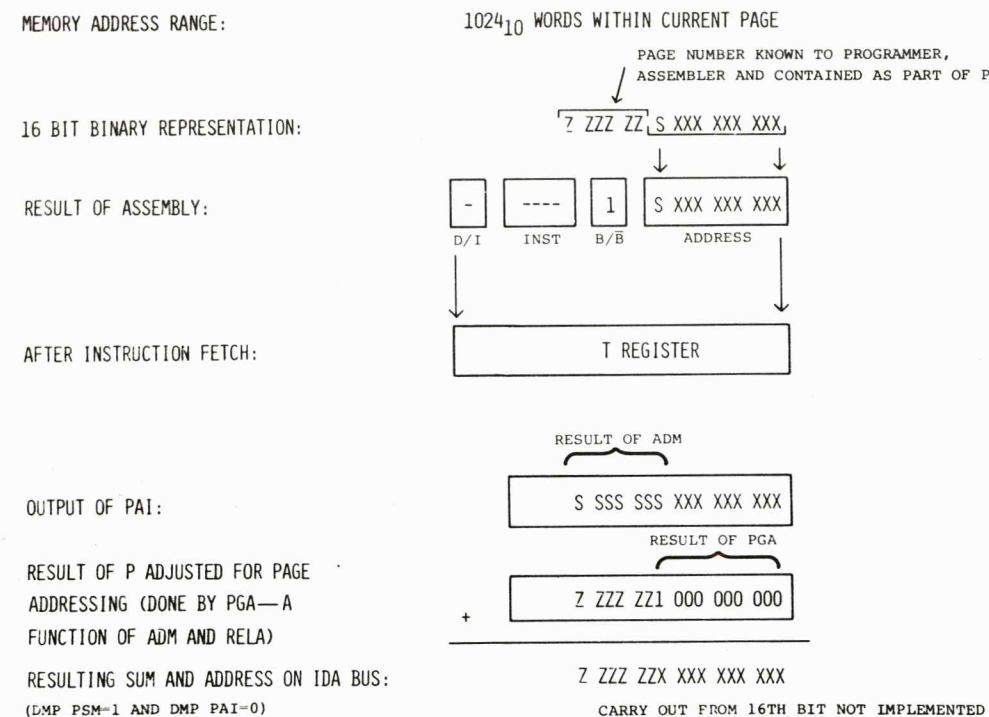


FIG 15-2-2-F

SECTION 15 (CONTINUED)

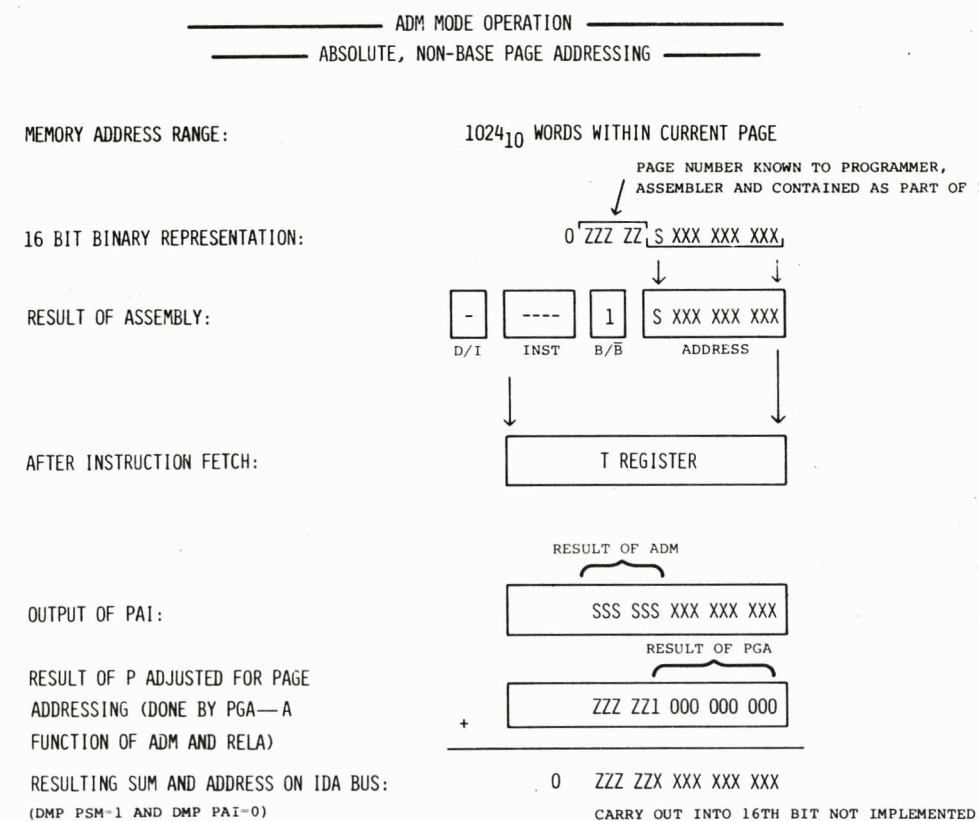
Figures 15-2-1 through 15-2-5 are illustrations of the various types of additions performed by the P-Adder mechanism. The drawings are really quite detailed, and we shall

spare ourselves a blow-by-blow description of each. The 15 and 16-bit versions are quite identical in their algorithms; they differ only in the presence of the extra bit.



D/I (DIRECT/INDIRECT) AND B/B (BASE PAGE/NOT BASE PAGE) CODED AS 0/1

FIG 15-2-3-S



D/I (DIRECT/INDIRECT) AND B/B (BASE PAGE/NOT BASE PAGE) CODED AS 0/1

FIG 15-2-3-F

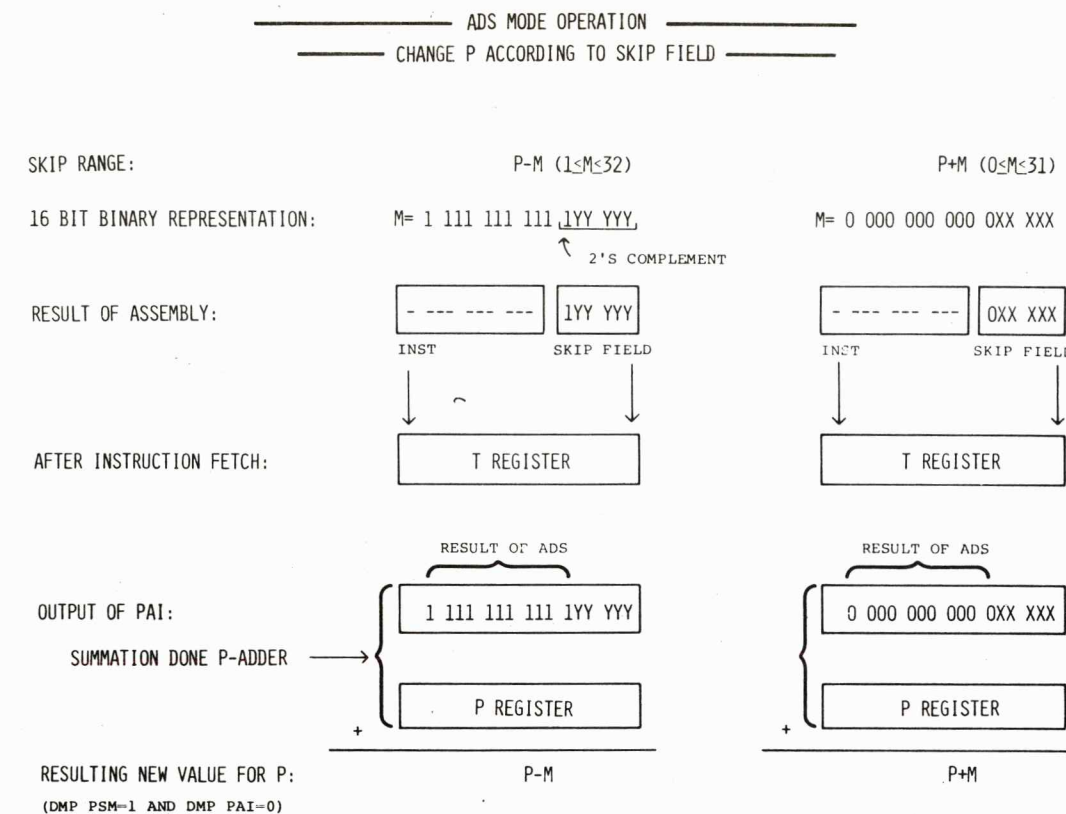


FIG 15-2-4-S

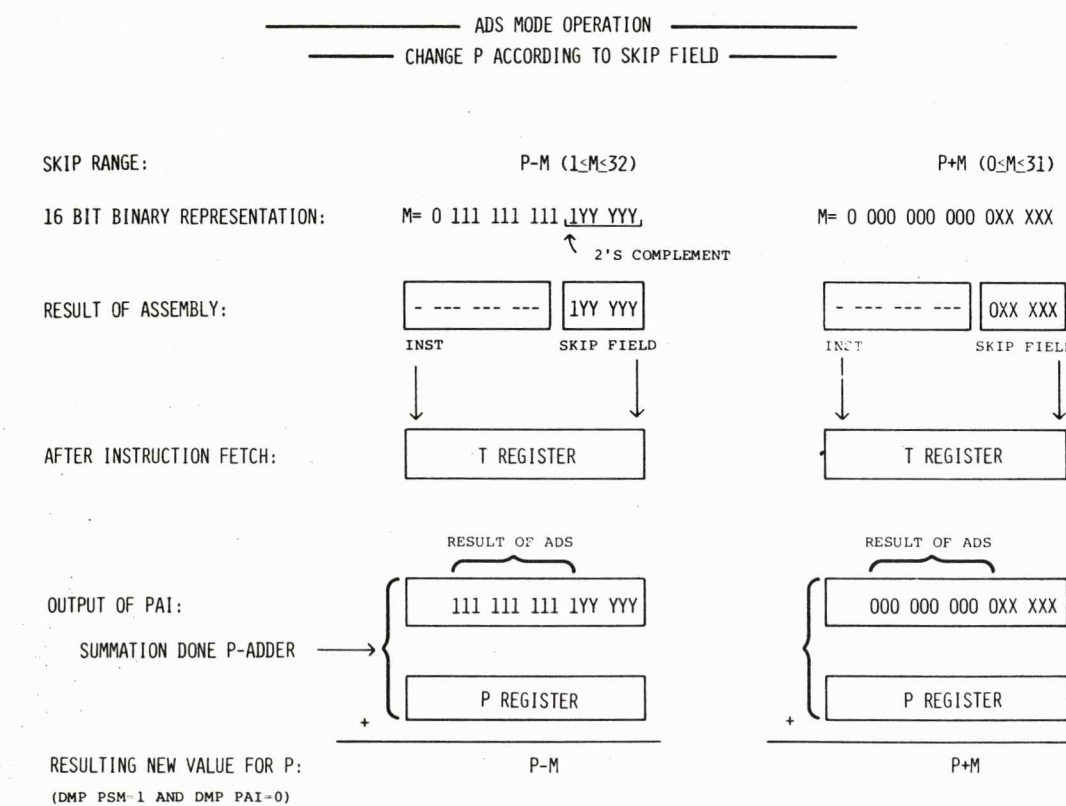


FIG 15-2-4-F

INCREMENTING P BY ONE

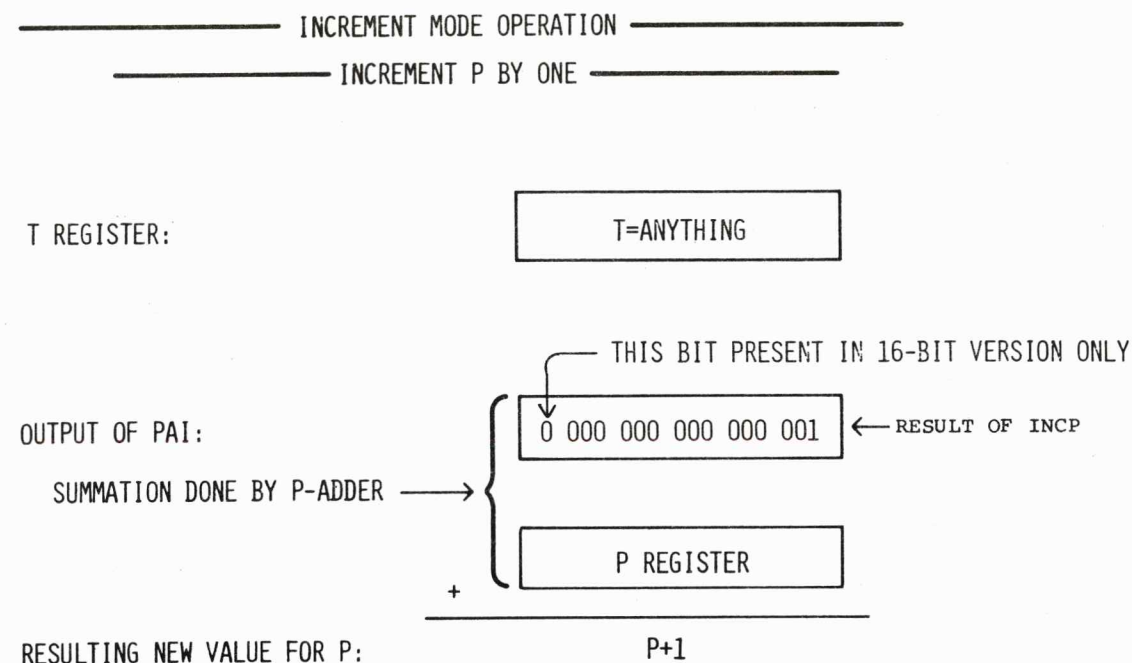


FIG 15-2-5

DETAILS OF THE PAI

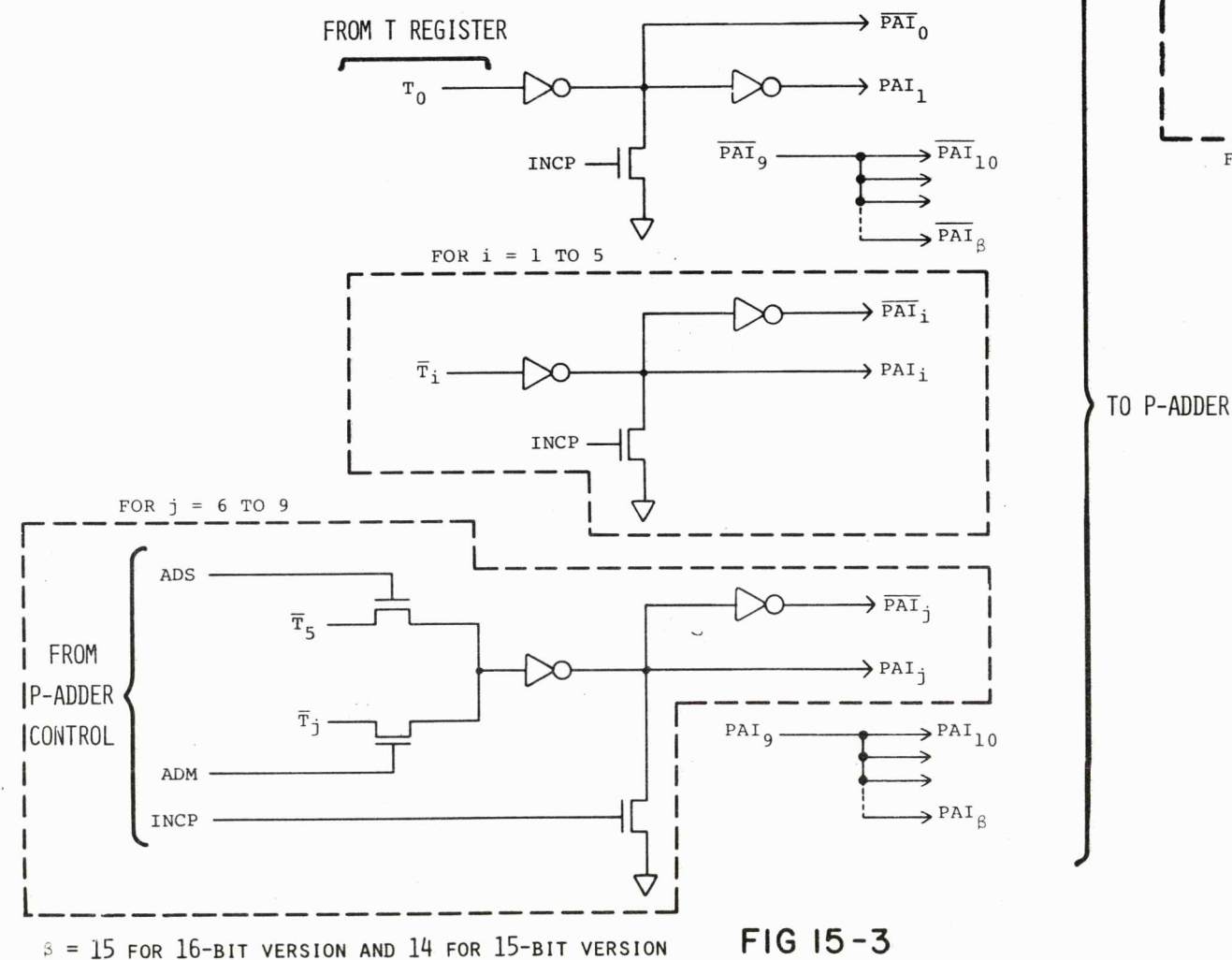


FIG 15-3

OVERVIEW OF THE P-ADDER

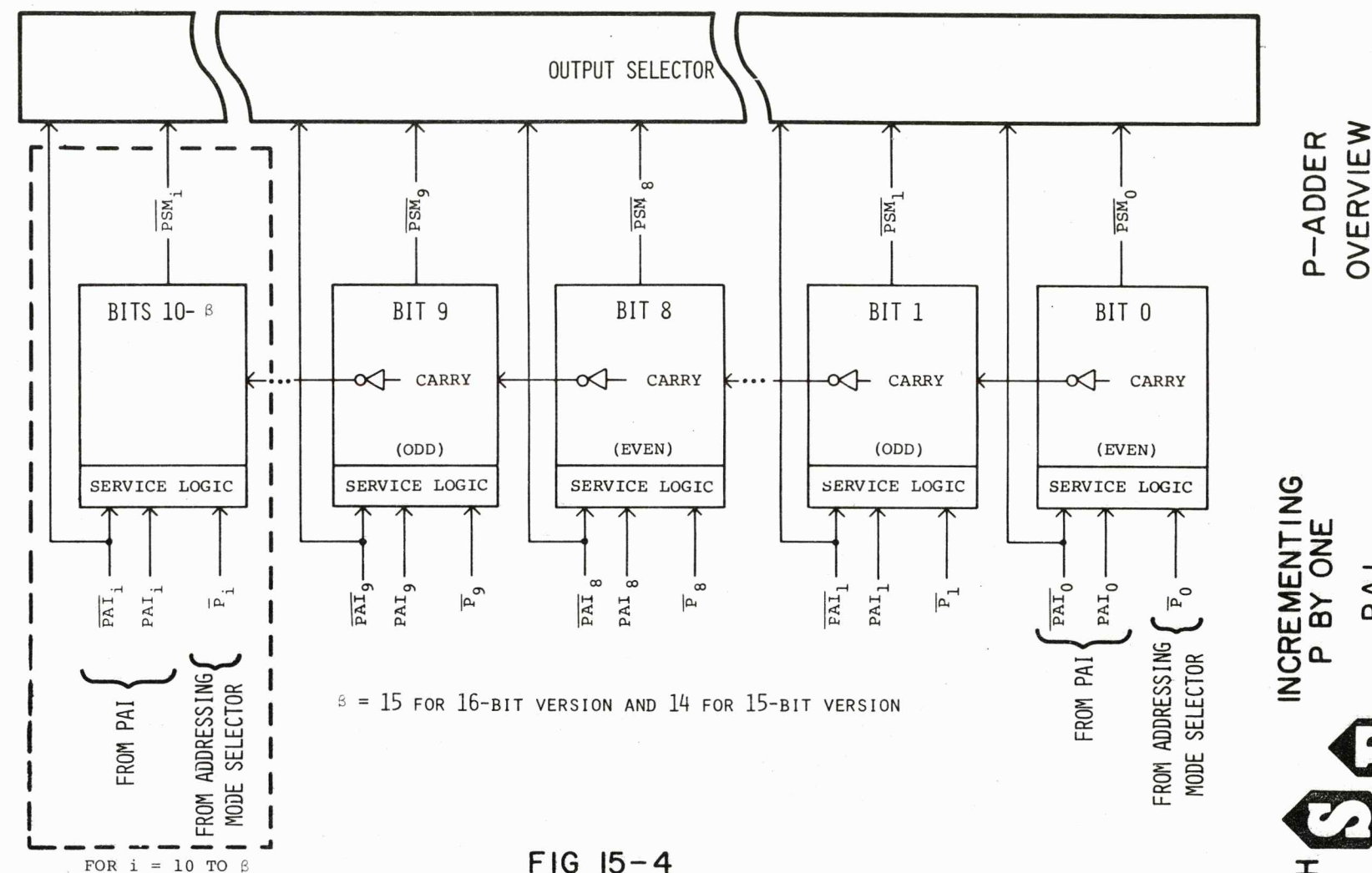


FIG 15-4

SECTION 15 (CONTINUED)

Figure 15-3 shows the details of the P-Adder Input circuitry. What the P-Adder Input circuit does is modify the value of T according to the type of add operation that is to be performed. During a normal program increment of the P register INCP is true and the output of the P-Adder Input is forced to a binary one.

ADS and ADM mode operation each generate results that are similar to each other. Operation in each of these modes requires leftward propagation of a sign bit. The sign bit will always be the leftmost bit in some number of bits that are right justified in the register. In the ADM mode it is bit 9 that is propagated to the left; in the ADS

mode it is bit 5. Such a propagation generates a two's complement number that can be added to the P register.

Figure 15-4 is an overview of the actual Adder mechanism. It is quite similar to the Adder in the ALU, except that there is no Complementer. The same general method of carry propagation involving even and odd numbered bits is employed. The nature and function of the Service Logic is identical. The inputs to the Adder are the outputs of the PAI and the contents of the P register, as possibly modified by the Addressing Mode Selector.

P-ADDER
OVERVIEW

INCREMENTING
P BY ONE
PAI

ADDRESSING WITH
SKIP FIELDS

ABSOLUTE
ADDRESSING

DETAILS OF P-ADDER CONTROL AND THE OUTPUT SELECTOR

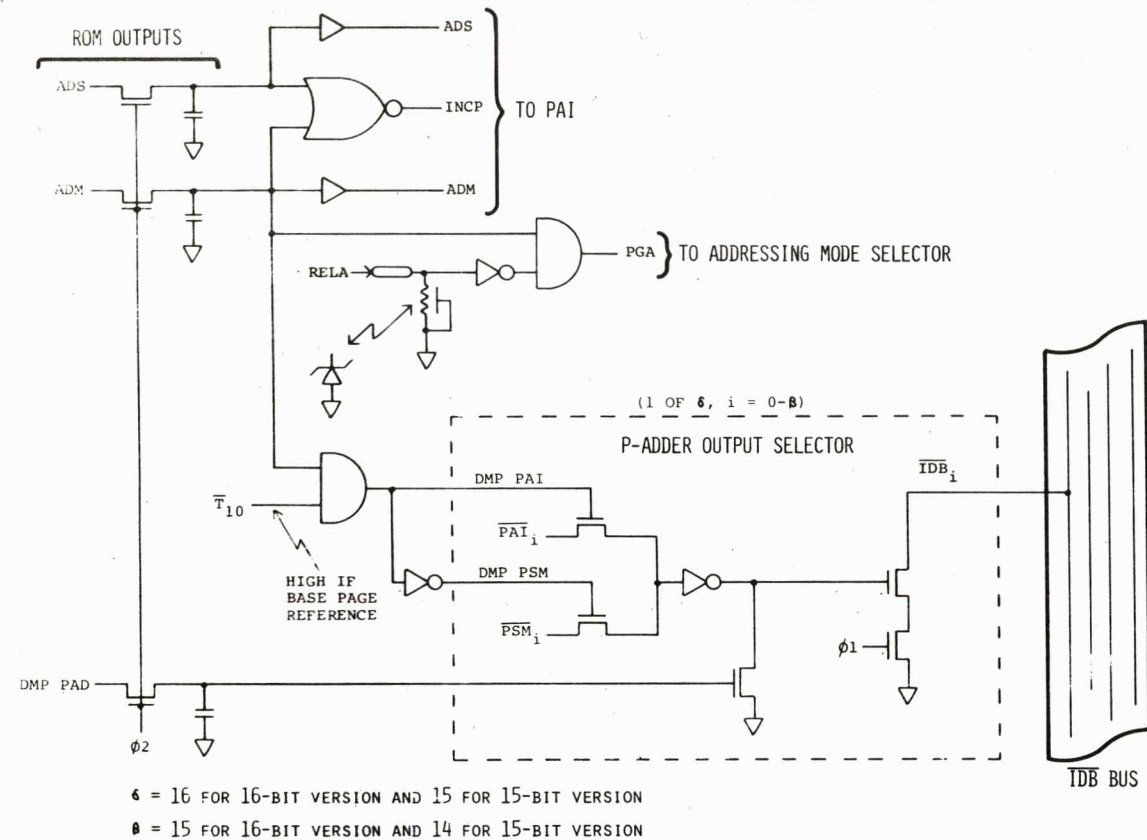


FIG 15-5

DETAILS OF THE ADDRESSING MODE SELECTOR AND SERVICE LOGIC

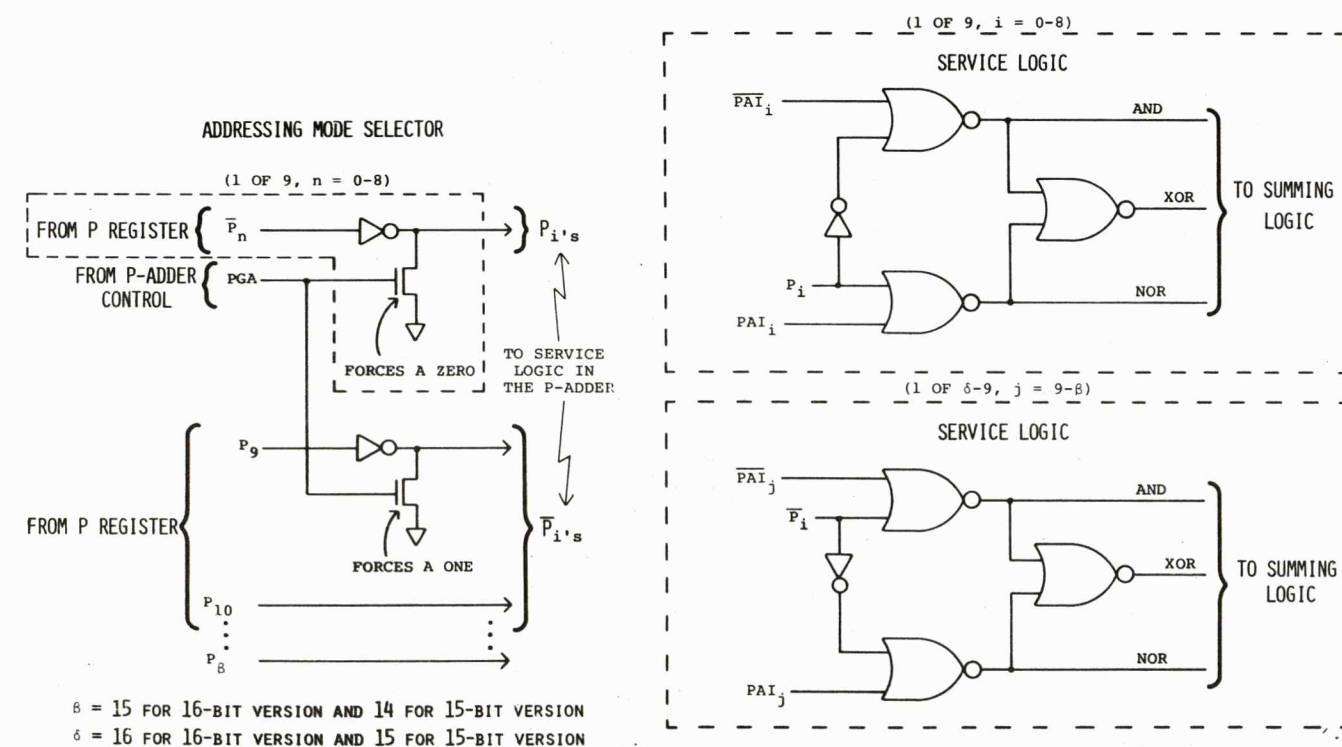


FIG 15-6

DETAILS OF THE SUMMATION AND CARRY PROPAGATION IN THE P-ADDER

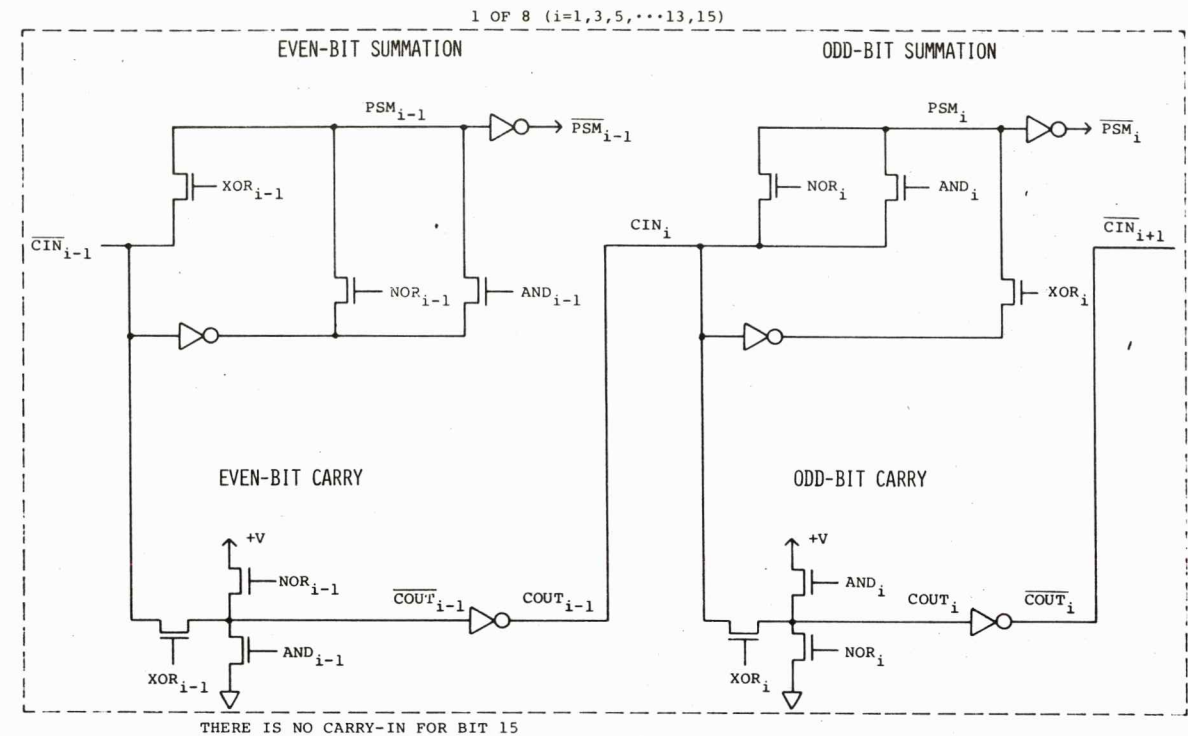


FIG 15-7

SECTION 15 (CONTINUED)

Figure 15-5 shows the circuitry of P-Adder Control and of the P-Adder Output Selector. One thing to keep in mind when examining this circuitry is that the micro-instructions ADS and ADM must be given continuously for the duration of the add operation. This is done by the flow charting in the ROM.

Figure 15-6 shows the details of the Addressing Mode Selector and the Service Logic used in the Adder. The Addressing Mode Selector adjusts the contents of the P register in the event that absolute page addressing is in effect. In such an event, it forces

the bottom nine bits of the P register to appear as zeros, while forcing the tenth bit to appear as a one. The upper bits remain unchanged. The effect of this is to produce an address in the middle of the page. At assembly-time the memory reference machine-instruction was given an operand chosen to be relative to that value.

The Service Logic portion is quite identical to that used in the ALU.

Figure 15-7 shows the details of summation and carry propagation in the P-Adder. This is no different than is done in the ALU.

OVERVIEW OF ADDRESS DECODE, INPUT PROTECTION.

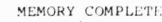


FIG 16-1

SECTION 16

Figure 16-1 is an overview of Address Decode and of the interrupt mechanism. The blocks labeled Input Protection, Data Switch and D register have already been discussed.

The purpose of Address Decode is to identify and latch memory addresses pertaining to registers within the BPC. Whenever a STM occurs the BPC Register Address Detector determines whether or not the memory address on the IDA Bus is one referencing a register within the BPC. If it is, then certain qualifiers and control signals are generated. At the same time, STM is used by the BPC Register LSB Address Latches to latch the four least significant bits of the IDA Bus. These bits will be used to determine which sets and dumps are necessary to properly respond to the memory cycle.

Also shown on this drawing is the connection between SET IDA and PDR. PDR is pulled low whenever the BPC is driving the IDA Bus.

Whenever a Memory Complete occurs the BPC latches the state of IDA(15). IDA(15) is the indirect bit, whether for machine-instruction bit patterns or for memory data. The Indirect Bit Latch generates a qualifier which may be used by the flow charting in the ROM to determine whether an indirect memory reference chain has been completed.

The general operation of the interrupt mechanism is as follows. If at the end of an instruction fetch $\overline{\text{INT}}$ is low, the Interrupt Grant Logic will arrange for the BPC to ignore the incoming instruction being fetched and replace that instruction with a JSM IV,I. It does that in the following way. First, the DMP

DETAILS OF THE INTERRUPT MECHANISM



FIG 16-2-S

IDA that would ordinarily bring the incoming instruction into the BPC from the IDA Bus is disabled. Next, an Interrupt Instruction Generator generates the bit pattern for JSM IV,I and places it on the IDB Bus. However, since that is an indirect memory reference instruction, and since the Indirect Bit Latch is driven by the IDA Bus and not the IDB Bus, the Interrupt Grant Logic must have its own private mechanism for setting the Indirect Bit Latch.

Figure 16-2 depicts the interrupt mechanism in detail. During an instruction fetch the ROM issues an INTE micro-instruction. INTE is AND'ed with INT to produce YINT. YINT is used to inhibit the DMP IDA which would ordinarily bring in the fetched instruction. YINT is also used as a command to generate the bit pattern for the machine-

instruction JSM IV,I.

The differences between the 15 and 16-bit versions involve the manner in which the Indirect Bit Latch is handled. In either case the Indirect Bit Latch is a simple latch whose output is $\overline{\text{IND}}$ (which is sent to the ROM), and whose input is active during MEC but inhibited during STP. In the 15-bit case the input to the Indirect Bit Latch is simply the OR between the interrupt request (INT) and the most significant bit of the IDA Bus. This is a scheme that lends itself not only to the needs of the interrupt mechanism, but also to multi-level indirect addressing. That is, there is *no restriction* on the ability of IDA(15) to set the Indirect Bit Latch each time there is a Memory Complete.

In the 16-bit case however, *restrictions are set on the manner*

DETAILS OF THE INTERRUPT MECHANISM

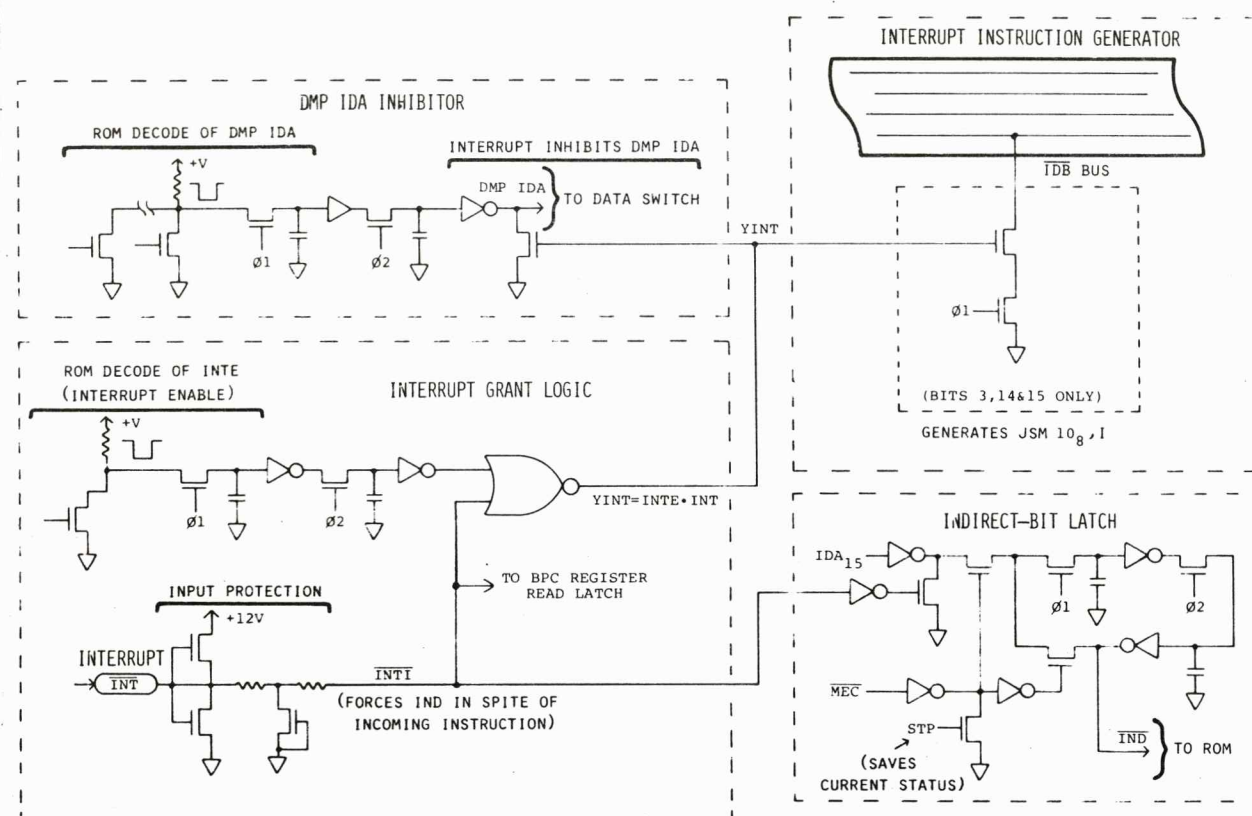


FIG 16-2-F

SECTION 16 (CONTINUED)

in which IDA(15) can set the Indirect Bit Latch. Such restriction is necessary to implement single level indirect addressing. The idea is to allow IDA(15) to set the Indirect Bit Latch only as a result of *fetching* an indirect memory reference machine-instruction. That is, the Indirect Bit Latch *cannot* be set by the occurrence of a one in bit 15 during memory cycles performed during the *execution* of an indirect memory reference machine-instruction. The way this is done is to AND IDA(15) with a signal that occurs *only* during instruction fetches. That signal is INTE. That logical product is then OR'd with the interrupt request line to form the input to the Indirect Bit Latch. (The ability of the interrupt circuitry to set the Indirect Bit Latch remains unaffected.)

Observe that STP is used to preserve the contents of the Indirect Bit Latch. STP will occur whenever

a Bus Grant is in progress. This is necessary, because a Bus Grant can occur anytime a memory cycle is not in progress. Also, it does not interfere with a Bus Grant, since there are no Bus Grant-related operations that could ever need to change the Indirect Bit Latch anyway. In the 15-bit version it is necessary to inhibit the latch, however, since memory cycles done during a Bus Grant could conceivably change the value of the latch. Inhibiting the latch in the 16-bit version is a redundancy since IDA(15) can only be loaded into the latch during INTE, and INTE is only given during the duration of the memory cycle, anyway.

One final comment. $\overline{\text{INT}}$ cannot be grounded indiscriminately. Refer to the various waveform diagrams for the BPC, and also for the IOC, for a description of how to ground $\overline{\text{INT}}$.

DETAILS OF BPC-REGISTER ADDRESS DETECTION

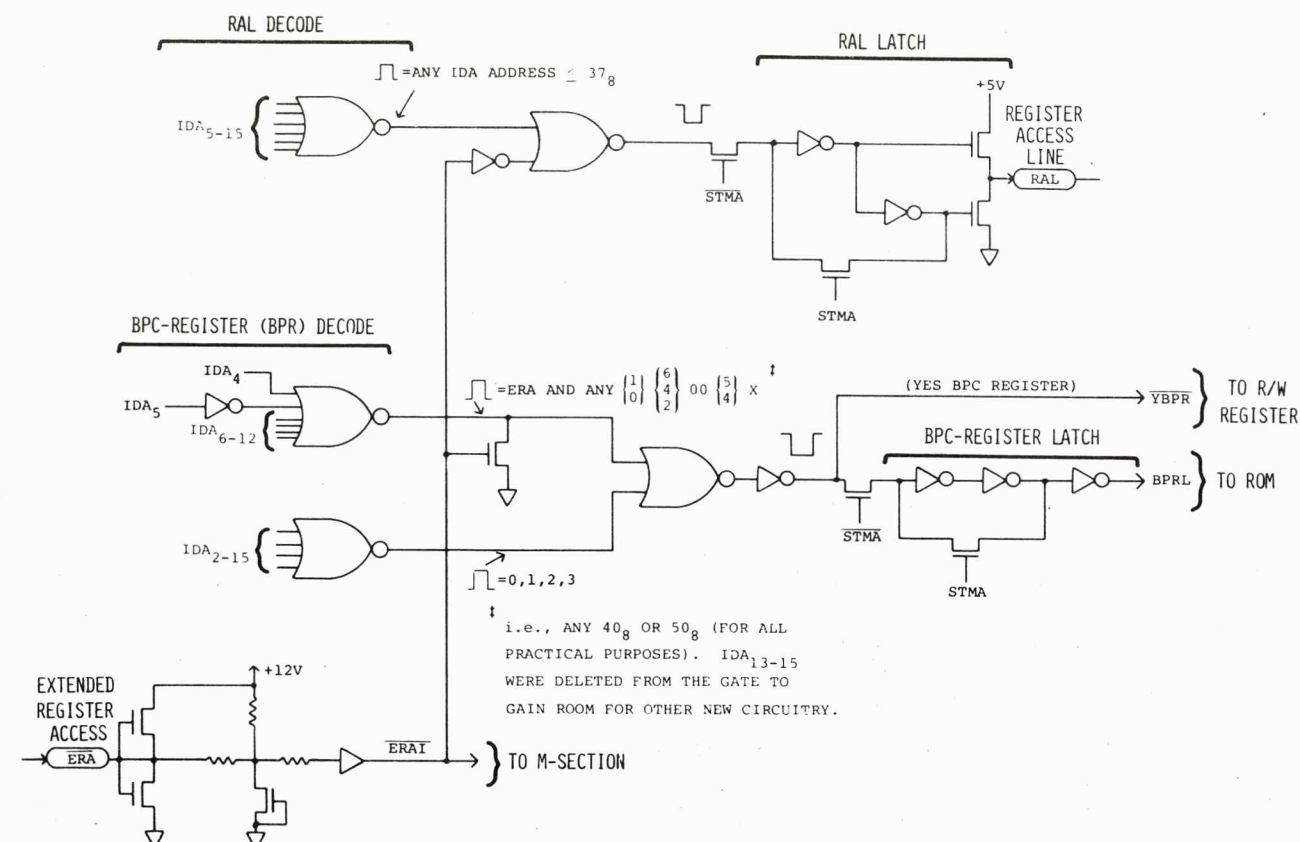


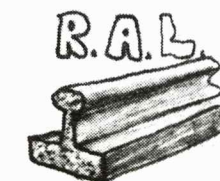
FIG 16-3-S

Figure 16-3 illustrates the details of the BPC Register Address Detection circuitry. To begin with, this circuitry generates a signal called RAL (Register Access Line). This is a service function that the BPC provides for all chips on the Bus. RAL means that a register address has occurred on the IDA Bus. Any memory address between octal zero and octal thirty-seven, inclusive, is a register address. It doesn't matter what the physical location of the register is; the BPC will generate RAL, regardless. In addition, use of the Extended Register Access mode will also generate RAL. RAL is latched by the occurrence of STM and lasts for the duration of the memory cycle. The occurrence of STM results in STMA (Start Memory Active). STMA is used to actually create the latch.

Two sets of gates are used to respond to register addresses associated exclusively with the BPC.

One gate responds to addresses zero, one, two and three, which represent the A, B, P and R registers, respectively. The other gate responds to the particular collection of ERA addresses assigned to locations within the BPC. The outputs of these two gates are OR'd together and latched by the BPC Register Latch. STMA is used to create this latch, also. The output of the latch is sent to the ROM as a qualifier.

The only difference between the 15 and 16-bit versions is that in the 16-bit version an extra bit must be included in the decoding of register addresses.



GROSSLY SIMPLIFIED FUNCTIONAL OVERVIEW OF THE M-SECTION

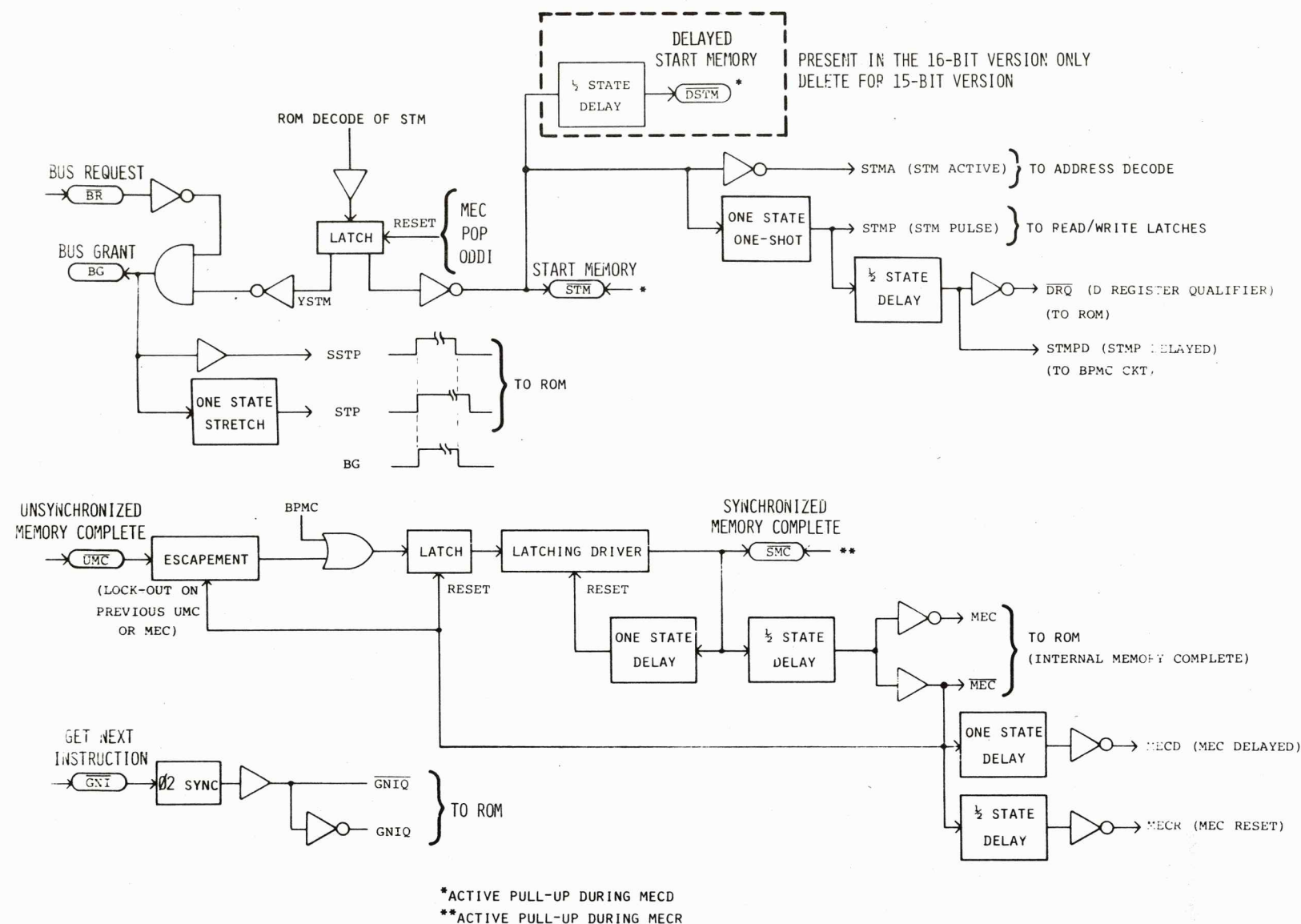


FIG 17-1-1

SECTION 17

Figure 17-1 is a greatly simplified functional overview of the M-Section. The best way to think of the M-Section is as an amorphous asynchronous collection of circuitry that responds to and controls memory cycles. Signals propagate through this circuitry in a manner more or less analogous to the way water soaks through a sponge.

Perhaps the best place to start is with STM. Whenever STM occurs, whether generated externally or by the BPC itself, a certain chain of events is put into effect. In the 16-bit version only,

a half-state after STM occurs a signal called Delayed Start Memory (DSTM) is issued. DSTM is intended for use with HP's 64K ROM. STM is also used to create STMA. A one-shot generates STMP. STMP is used by the Read Register and Write Register Latches. After a further delay STM generates DRQ (D Register Qualifier) and STMPD (STMP Delayed).

DRQ is not easy to explain. There are occasional instances in the flow charting where, in response to memory cycles directed to registers within the BPC, a state containing a SET D/SET IDA is accessed two times in a row. The purpose

for doing this is to get the two consecutive SET IDA's in establishing an address on the IDA Bus. However, if a second SET D is also allowed to happen it can create a glitch on that address. DRQ is used in decoding the SET D; the delayed appearance of DRQ prevents the decoding of the second SET D.

STMPD is used to generate Memory Complete, in the event the memory cycle was to a register contained within the BPC.

The BPC itself can also generate STM. A single instance of decoding STM from the ROM sets a latch which then creates a steady

state Start Memory, until the latch is reset by MEC, POP or ODD. The latch also generates YSTM, which is used to inhibit the issuance of a Bus Grant. The usual means for resetting the STM latch is by a Memory Complete at the end of the memory cycle (i.e., via MEC).

Next, consider the various Memory Complete signals. The easiest of these to use is UMC (Unsynchronized Memory Complete). It can be used without regard for when its leading edge occurs, and without regard for the signal's duration. UMC can be very long or very short. An escapement mechanism requires that a previous UMC be released before a new UMC can be effective. The escaped UMC is OR'ed with BPMC to set a latch. This latch will be automatically reset by MEC. BPMC is given whenever the BPC generates its own Memory Complete. This happens in response to memory cycles where the destination is a register within the BPC.

The above mentioned latch sets a latching driver whose output is SMC (Synchronized Memory Complete). The latching driver automatically resets itself after a one-state delay. SMC is the signal actually used by the various memory entities to understand that a memory cycle is complete. SMC is further delayed internally to generate MEC. MEC goes to the ROM to indicate that the memory cycle is complete. MEC itself is further delayed to produce MECD and MECD. These are used as various reset functions within the M-Section itself.

A Bus Request will automatically generate a Bus Grant, provided that there is no ongoing memory cycle, as indicated by YSTM. Bus Grant is used to generate SSTP and STP. These signals have coincident leading edges, but SSTP is of shorter duration than STP. The reason for this is as follows. When a Bus Grant occurs the activity in the ROM must be suspended. The vast majority of micro-instructions in the ROM are decoded against STP; when STP occurs they vanish. Suppose a Bus Grant occurred just at

GROSSLY SIMPLIFIED FUNCTIONAL OVERVIEW OF THE M-SECTION

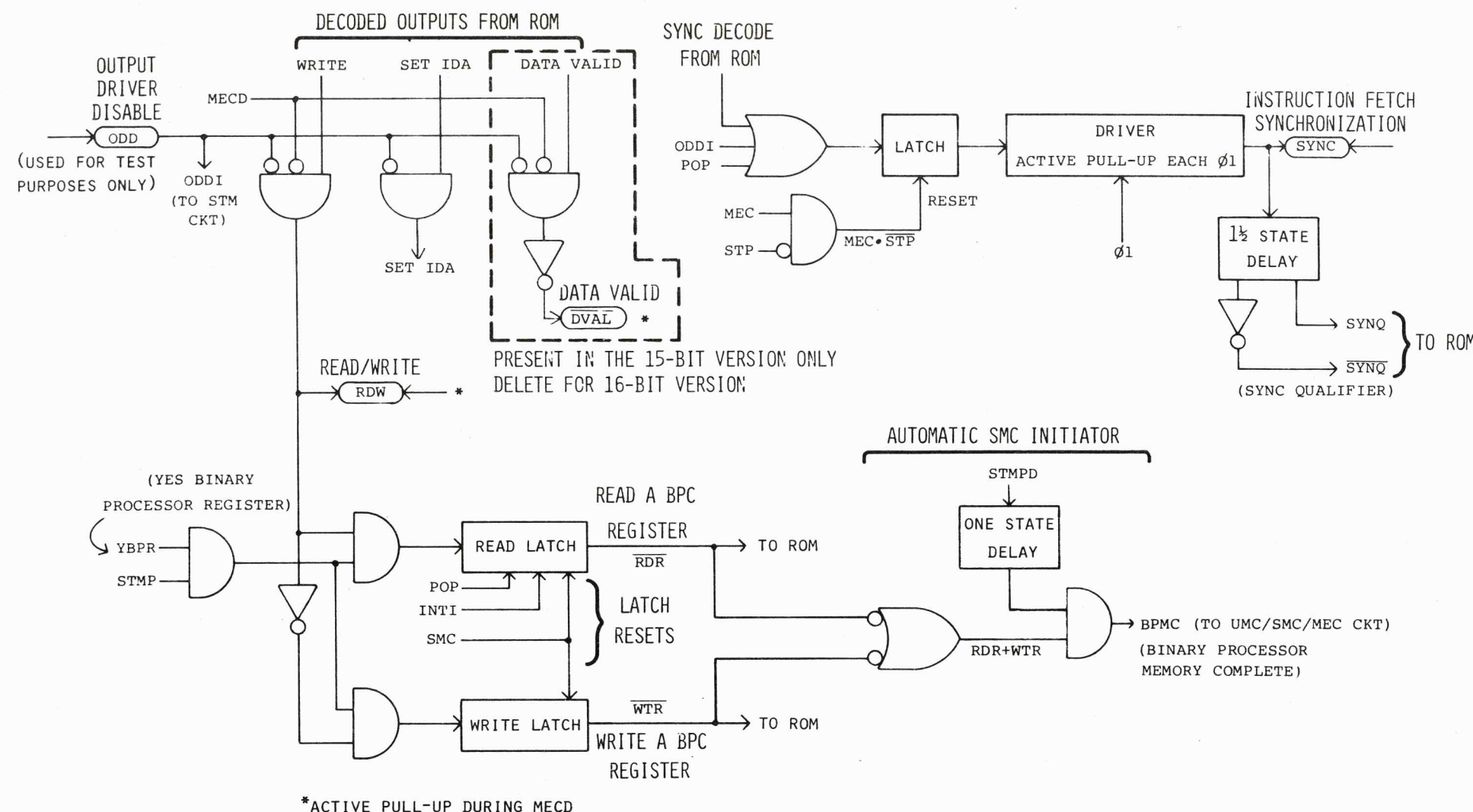


FIG 17-1-2

SECTION 17 (CONTINUED)

the very beginning of a memory cycle, but before STM has been given; that is, after the first SET IDA but before the second SET IDA (i.e., in the middle of sending out the address). In such a case, Bus Request would generate Bus Grant, since the memory cycle is not yet properly underway. After the Bus Grant is over it is not possible to simply resume execution of the flow charting. It is necessary to repeat the first SET IDA. But that point in the flow charting has already been passed. The solution is to issue the second SET IDA twice. This is done by having SET IDA decoded against SSTP, while everything else is decoded against STP. In this way SET IDA is given a headstart, compared to the other

micro-instructions in the ROM, and the SET IDA/SET IDA STM convention can be preserved.

GNI is part of a mechanism that allows the BPC to be compatible with other chips in the system that might possibly require two-word instruction fetches. At present, no such chips exist, nor is it likely that there ever will. However, the possibility has been allowed for. The mechanism is further explained in the flow charting for the ROM.

The Read/Write line (RDW) indicates if a memory cycle is to be a read memory cycle or a write memory cycle. If Address Decode determines that a memory cycle concerns a register within the BPC the state of RDW is latched for

later use by the ROM. This is done by using the AND of YBPR and STMP to enable a separate latch for each of the read and write conditions. Each latch is reset by SMC, and also at turn-on. Two latches are used so that neither qualifier to the ROM is active when no memory cycle is in progress.

Observe that an interrupt resets the Read Latch. This is done in the event that the aborted instruction fetch was occurring from a BPC register. Also notice that POP does not reset the Write Latch. It simply turns out that it is never necessary, and that circuitry can be eliminated. (During the turn-on sequence the status of the Write Latch does not matter, and the first

instruction fetch resets it anyway.)

In addition to being sent to the ROM as qualifiers, the outputs of the Read and Write Latches are OR'ed together and then AND'ed with STMPD to produce BPMC. BPMC is the means whereby the BPC can cause an SMC on its own initiative.

Observe that the decoded ROM outputs for SET IDA, Data Valid and Write can be rendered impotent by a signal called ODD (Output Driver Disable). Also notice that Write is not a latched ROM output. Write must be continuously decoded from the ROM for the duration of the memory cycle.

An important difference between the 15 and 16-bit versions is that Data Valid is totally absent from the 16-bit version.

The SYNC circuitry forms an important part of the M-Section. SYNC can be given any of three ways: as a decoded ROM output; as a result of ODD; or, as a result of POP. Any of the three sets a latch which is reset only by a Memory Complete not occurring during a Bus Grant. Since SYNC is normally given only when there are no more memory cycles to be performed as part of the execution of a machine-instruction, this means that the memory cycle that resets SYNC is the next instruction fetch. Indeed, the purpose of SYNC is to indicate that the next non-Bus Grant memory cycle is an instruction fetch.

SYNC itself is not a steady-state signal. It is pulled up each phase one. During the next phase two SYNC is either left ungrounded if it's to be high, or pulled down to ground if it's to be false. Thus, SYNC should be looked at only during phase two. The phase two state of SYNC is detected and then delayed to be sent to the ROM as a qualifier. This qualifier is used by the BPC to determine if all the chips on the IDA Bus agree that SYNC should go high, and that the next instruction fetch should proceed.

M-SECTION
OVERVIEW

M-SECTION
OVERVIEW

DETAILS OF RDW, THE READ AND WRITE LATCHES, AND THE AUTOMATIC SMC INITIATOR

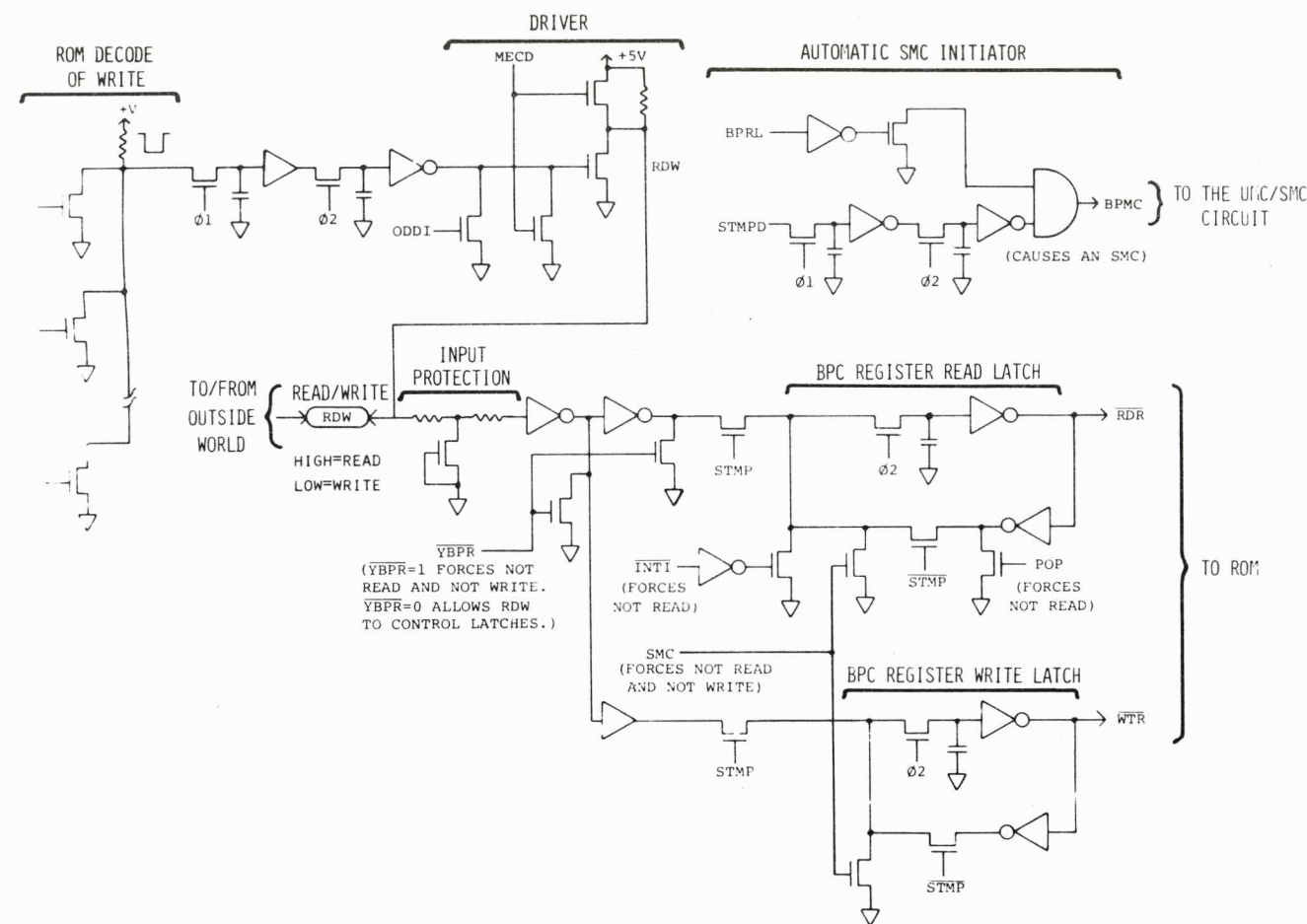


FIG 17-5-S

DETAILS OF RDW, THE READ AND WRITE LATCHES, AND THE AUTOMATIC SMC INITIATOR

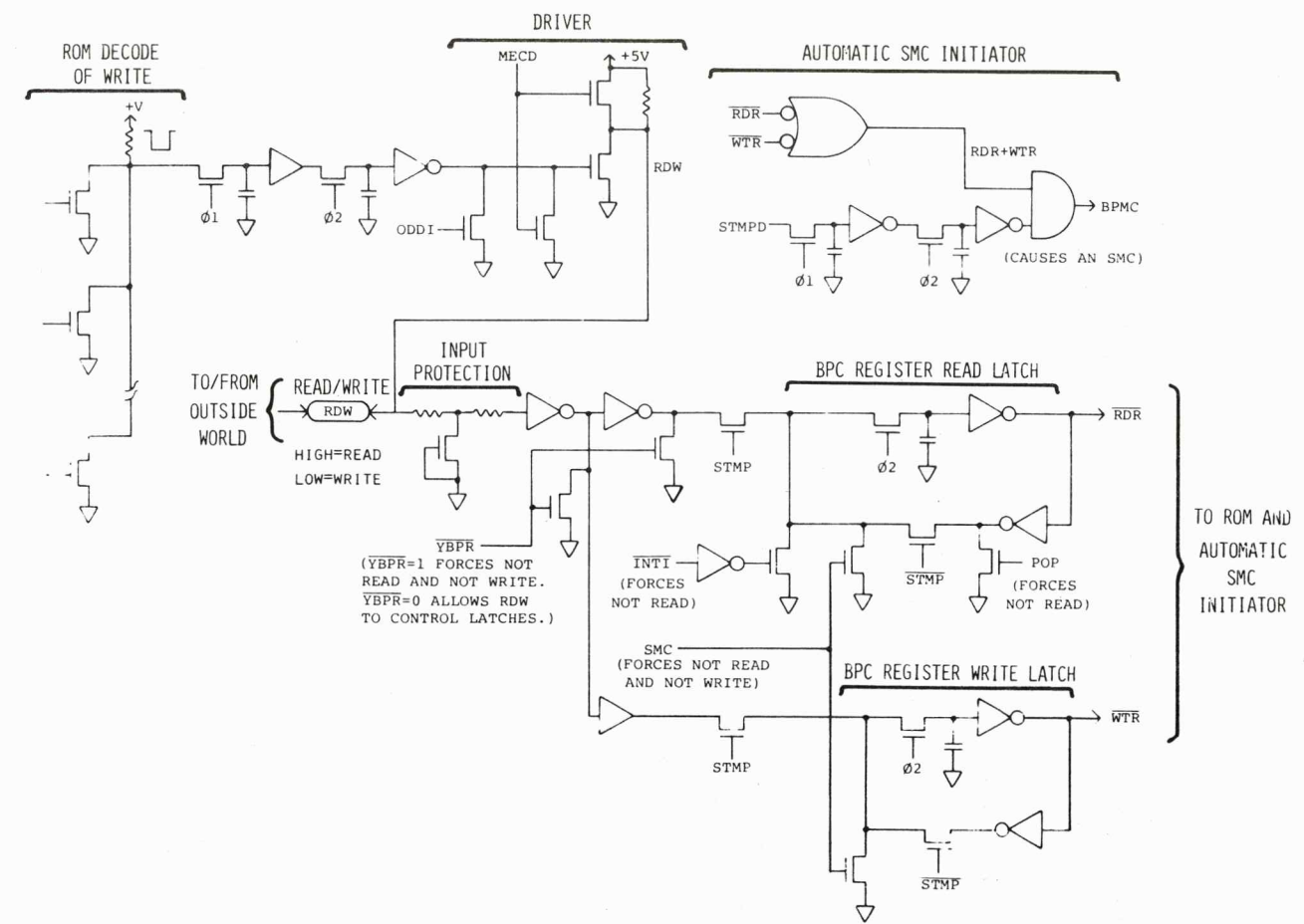


FIG 17-5-F

SECTION 17 (CONTINUED)

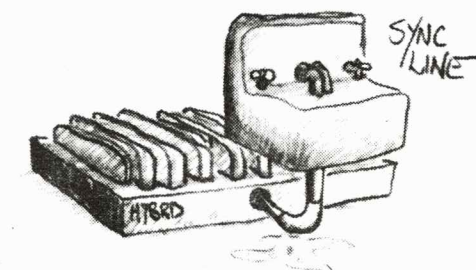
Figure 17-5 illustrates the circuitry controlling the RDW line, the BPC Read and Write Register Latches and the Automatic SMC Initiator. The circumstances surrounding the BPC Register Read and Write Latches have already been explained. There is a significant difference between the 15 and 16-bit versions concerning the Automatic SMC Initiator. It amounts to this: In the 15-bit version BPMC was incorrectly generated. The problem concerns instruction fetches from BPC registers, such as might be encountered subsequent to an EXE A or JMP A instruction. If an interrupt should occur during the resulting instruc-

tion fetch from the referenced BPC register, the BPC will fail to give an SMC. This brings all system activity to a screeching halt. The problem exists because the outputs of the Read and Write BPC Register Latches are used to initiate the SMC via BPMC. In the case of an instruction fetch from a BPC register it is the Read Register Latch that is set. Unfortunately, it is also the Read Register Latch that is reset by an interrupt. That reset occurs before BPMC is generated. There is no cure for this, other than to shut off the interrupt system during periods of time when instructions might be fetched from

BPC registers.

Note that the problem does not exist for instructions such as EXE A,I or JMP A,I since these do not cause an instruction fetch from A; they only do a read of A. It is only during an instruction fetch that an interrupt can occur.

The problem was fixed in the 16-bit version by using BPRL to initiate BPMC. This works perfectly well because BPRL is a latched output from address decode but is not reset by interrupt. Also, anytime BPRL is true YBPR is true, and that is the same thing that's used to enable the BPC Register Read and Write Latches.



DETAILS OF THE SYNC CIRCUIT

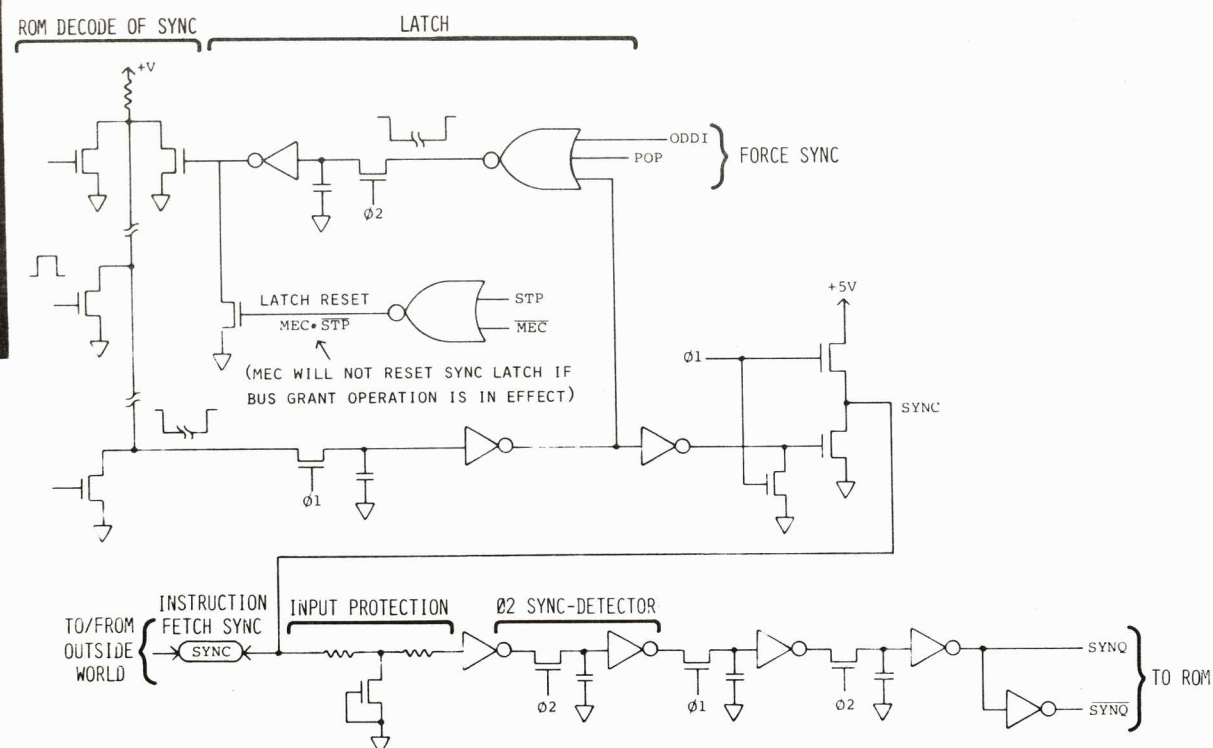


FIG 17-6

DETAILS OF THE DVAL, ODD CIRCUITS, SET IDA DECODE CIRCUITS, AND PDR

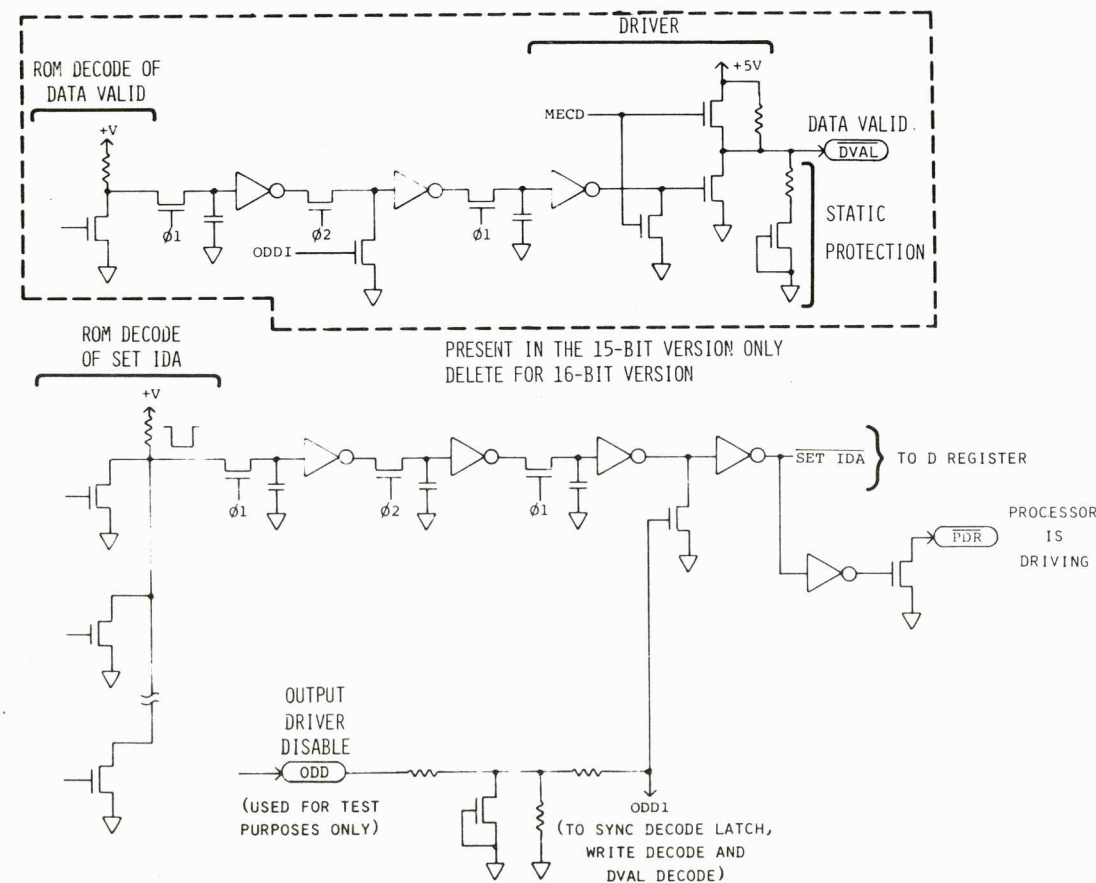


FIG 17-7

DETAILS OF THE BUS REQUEST/GRANT AND GNI CIRCUITS

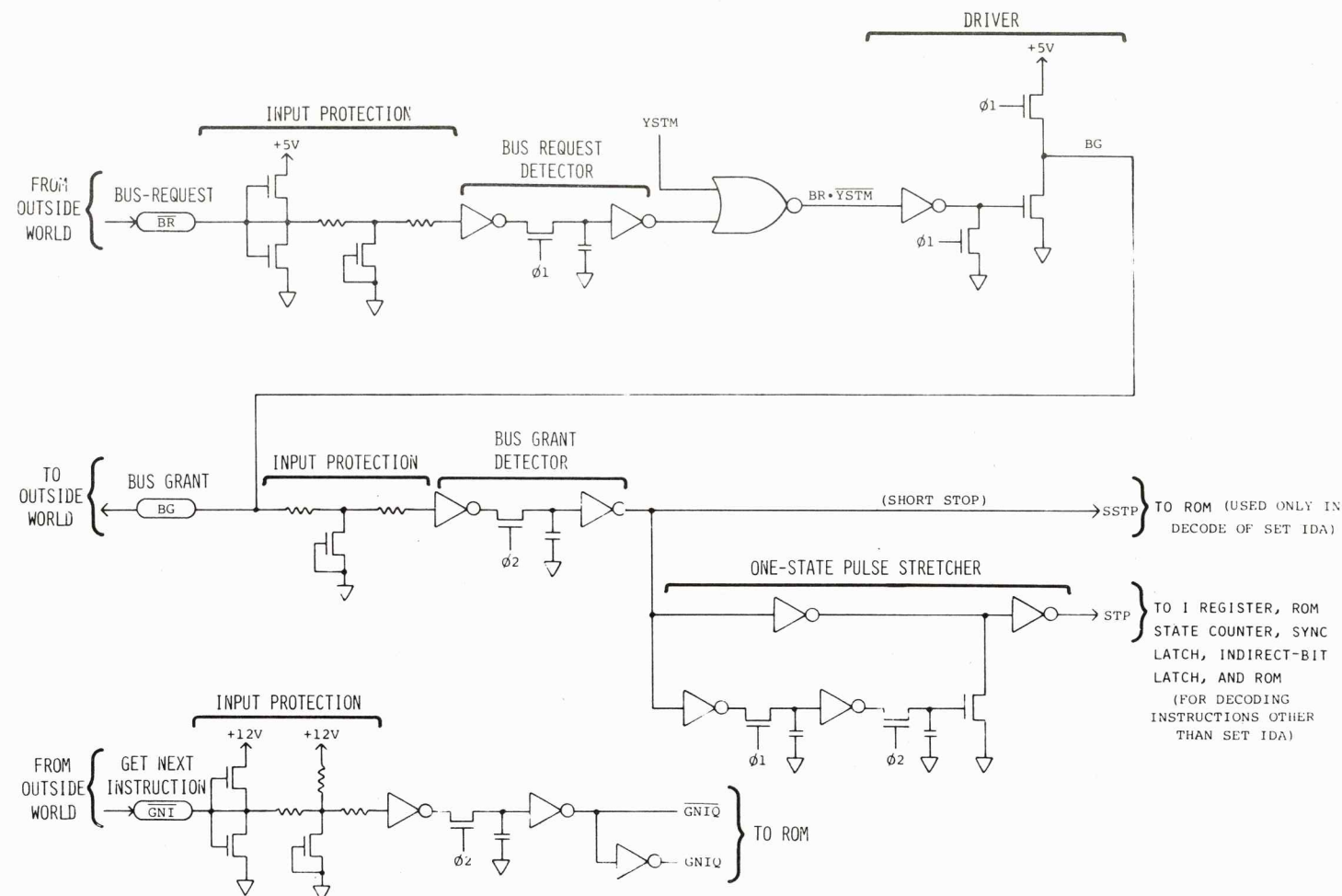


FIG 17-8

SECTION 17 (CONTINUED)

Figure 17-6 shows the details of the SYNC circuit. Observe the latched ROM output that can be both reset and forced by various signals. Also note that SYNC is pulled up during each phase one. Since SYNC is truthful only during phase two, the phase two SYNC detector should not be considered a delay, as much as a means to connect the delay and qualifier driver circuitry for SYNC to the SYNC line only during those times when SYNC is truthful.

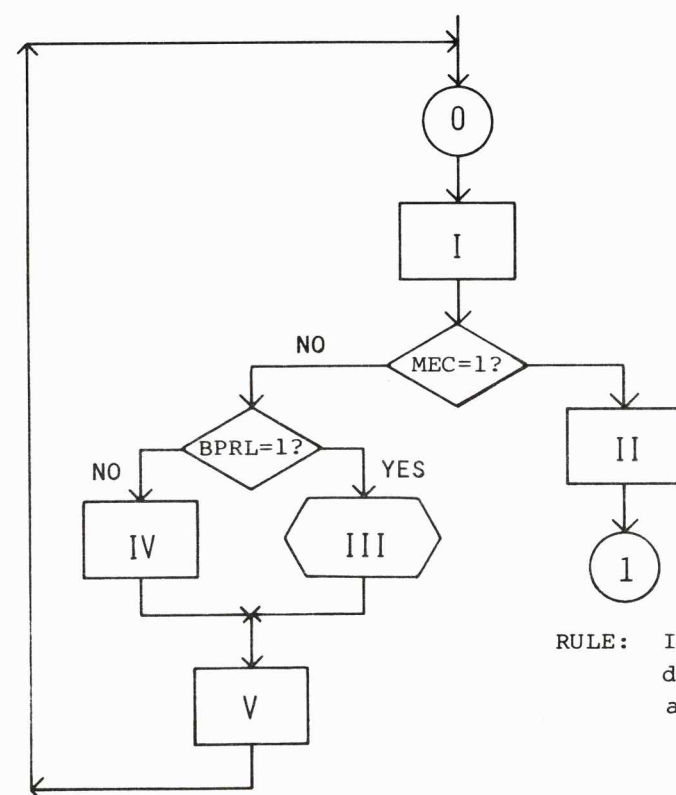
Figure 17-7 illustrates the details of the Data Valid, ODD, SET IDA and PDR circuits. This stuff is pretty much as has been described earlier. The Data Valid circuit was deleted from the 15-bit version to provide room for the Delayed Start Memory circuit on the 16-bit version.

Figure 17-8 shows the details of the Bus Request, Bus Grant and GNI circuits. The Bus Request line is normally precharged by the IOC on phase two. Therefore, the Bus Request Detector looks at Bus Request only during phase one. Next, Bus Request is AND'ed with the absence of STM and used to generate Bus Grant. Notice that Bus Grant is always precharged on phase one and pulled down on phase two if necessary. The Bus Grant Detector looks at Bus Grant each phase two and responds to a *successful* Bus Grant. (Even though the BPC allows Bus Grant, some other device *might not* allow it by keeping Bus Grant grounded.) A detected Bus Grant is used to generate SSTOP and its longer counterpart, STP.

The GNI circuit looks at GNI during phase two and produces a stable qualifier for use in the ROM during phase one.

HOW TO INTERPRET THE BPC ASM CHART, CONT.

1. What we usually refer to simply as a state ("state 4 for LOAD A") is generally a coincidence of that particular state-count and some group encoding qualifier pattern (representing a particular group). The most precise way to refer to a location on the ASM chart is indicate both the group and state-counts, (say, B4). Some states (0,1, and 14) are completely group independent. States are indicated by circles with numbers in them: (4). Group information is prominently displayed next to sections to which it pertains.
2. Each state represents a $\phi 2$ pre-charge and $\phi 1$ decode in the ROM. The ASM chart represents what is decoded from the ROM in the various states; it does not necessarily represent end-results that occur simultaneously. If, for instance, two instructions decoded in the same state have different delays coming from the ROM, then they do not result in simultaneous activity, even though they are drawn as being in the same state.
3. Rectangular boxes (SET A) denote micro-instructions. Diamonds (MEC=1?) denote qualifiers affecting the decoding of micro-instructions. Ovals (STM) denote micro-instructions that are actually decoded and given, but that are "don't-cares". That is, they are present but do not affect the algorithmic process. Sometimes these don't-cares are the result of minimization, and sometimes they are a result of the way the flow charts have been drawn (in an attempt to make them more easily understood).
4. All activity within a state is decoded and initiated (its delay is begun) at the same time. The fact that a state is shown as a sequential arrangement of boxes and diamonds does not imply sequential activity; the entire state is decoded simultaneously. For example, state 0 is represented below:



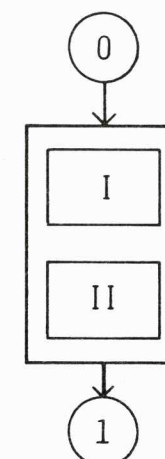
RULE: Identify the path, then decode all instructions at once.

FIG 18-1-1

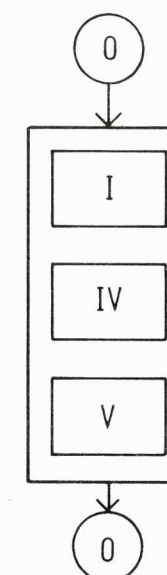
HOW TO INTERPRET THE BPC ASM CHART, CONT.

Another way to represent the same activity is illustrated below. We don't draw the ASM chart that way because of the increased size and because of problems in achieving connectedness. Also, overall algorithmic process would be hard to see; the more compact notation results in a more effective visual outline. Within a state however, the expanded notation is often less confusing as it more closely represents the actual way things are done.

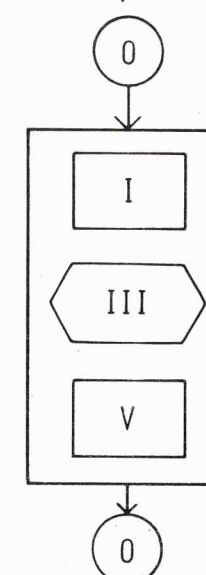
If MEC=1, then:



If MEC=0, and BPRL=0, then:



If MEC=0 and BPRL=1, then:



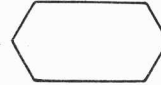
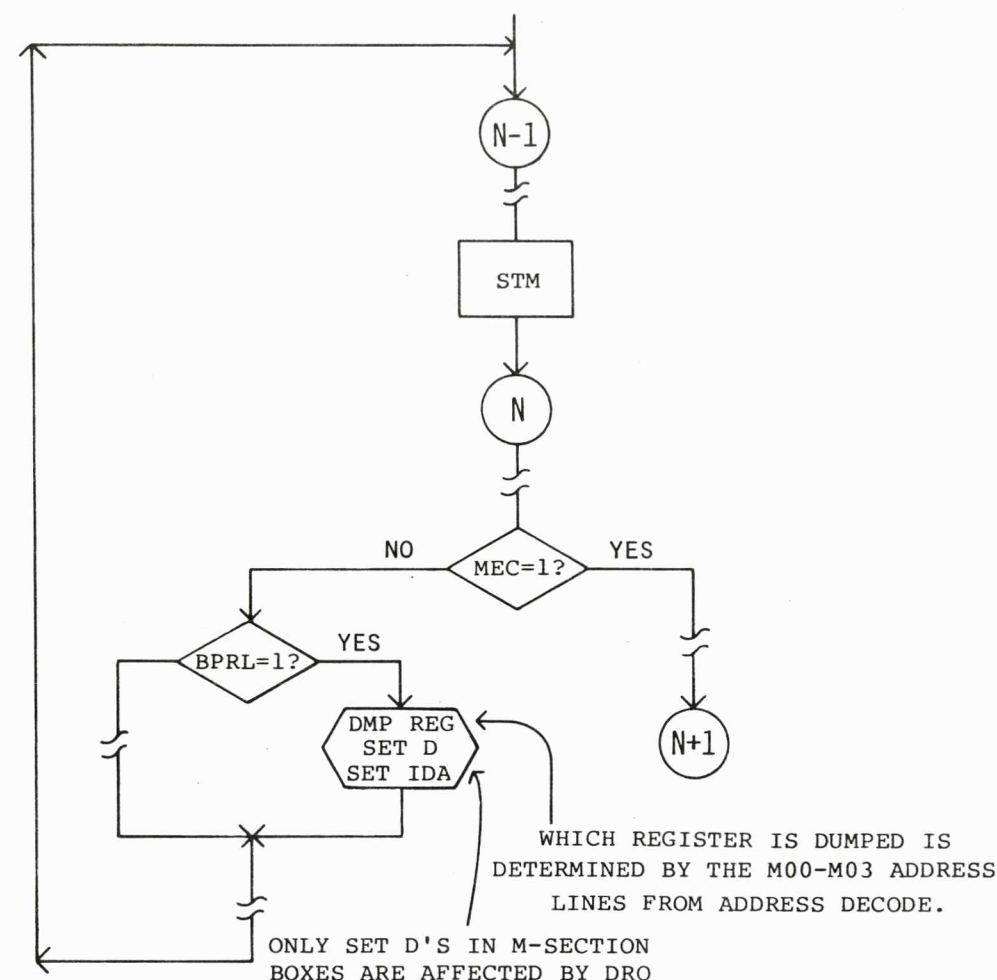
5. Within a state, related instructions are grouped together in the same box solely for the sake of algorithmic clarity.
6. Because of limitations on transistor device size, and the large capacitances of the IDA lines, two consecutive SET IDA's are required to ensure that IDA lines assume their proper final values. If what is being transmitted with the SET IDA is an address for Memory, the STM will accompany the second SET IDA. If data is being written to Memory, DVAL will accompany the second and all subsequent SET IDA's until Memory Complete is received. In either case, the IDA lines will be stable before the start of the second SET IDA.
7. The symbol  represents activity controlled by the M-Section. The micro-instructions shown inside are encoded in the ROM, just as are any other micro-instructions. However, these particular instances of decoding those micro-instructions are independent of all group and state-count qualifier lines in the ROM. They are decoded against qualifiers generated by the M-Section and Address Decode. (\overline{RDR} , \overline{WTR} , \overline{DRQ} , M00-M03).
8. The qualifiers that enable such M-Section activity are generated when STM occurs in conjunction with an address on the IDA lines that specifies a register within the BPC. These qualifiers are not always generated immediately, nor are they necessarily co-incidental with one another.

FIG 18-1-2

SYNC CIRCUIT
 DVAL, ODD, SET IDA
 DECODE, AND PDR
 BUS REQUEST,
 BUS GRANT, GNI
 ASM CHART
 INTERPRETATION
 ASM CHART
 INTERPRETATION (CONT)

HOW TO INTERPRET THE BPC ASM CHART, CONT.

9. Consider the following typical situation:



In this example the STM occurs in the state prior to the one with the M-Section activity. The machine will stay in state N for 4 consecutive state-times: 3 "no's" and a "yes" for the MEC qualifier. The BPRL qualifier (from Address Decode) will be met each time, but due to delays in the M-Section the first pass through state N generates none of the M-Section activity. The second and third passes do perform the indicated activity, except for the SET D. It is done during the second pass, but not during the third. This is a result of \overline{DRQ} , and prevents D from tracking the pre-charge of the \overline{IDB} Bus which follows the execution of the SET D. This prevents the second SET IDA from producing a glitch on the IDA lines.

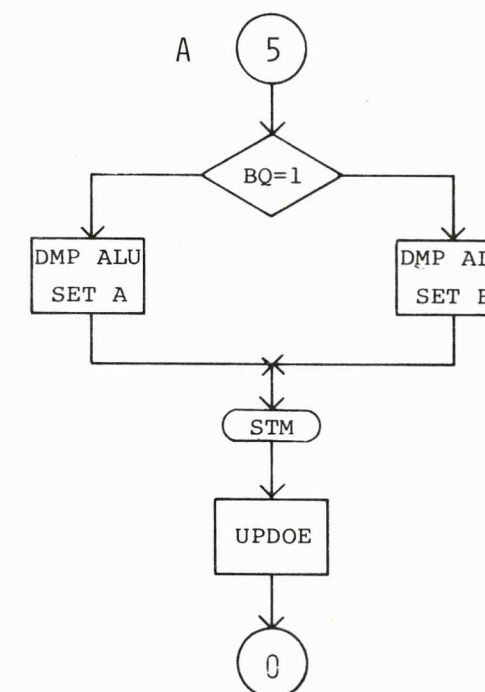
Sometimes the STM is given in state N-2. In such a case the one state of M-Section delay is spent while in state N-1, and only three state-times are spent in state N. When this happens the M-Section activity occurs immediately on the first and second of these; the third meets the MEC qualifier. As before, the SET D is done only once.

Such an instance occurs between states C2 and C9. There are several others.

FIG 18-1-3

HOW TO INTERPRET THE BPC ASM CHART, CONT.

10. Sometimes, as in state C9, the BPRL qualifier is shown by a dotted line: $\langle BPRL=1? \rangle$. This occurs only when the machine represented by the ASM chart has no activity in that state that is conditional upon BPRL. Therefore, BPRL is a don't-care for the ASM chart at that state, and indeed is not used in the ROM there. However, it still initiates M-Section activity when met. And when it is met no externally caused MEC will be forthcoming, since no memory external to the BPC is involved. The MEC will be supplied by the M-Section itself, as soon as its activity is finished. In these cases the timing is as outlined in 9 above, and we show a BPRL qualifier that affects the M-Section, but not that point in the ROM, to remind the reader of what's happening.
11. Finally, a word about the correspondence between what's in the ROM and how the flow charts are drawn. The flow charts have been "de-minimized" to promote their ease of understanding. For instance, state A5 is shown as:



There are not two instances of decoding DMP ALU for state A5. The decoding of DMP ALU in that state is independent of the BQ qualifier, as it is done regardless of that qualifiers outcome.

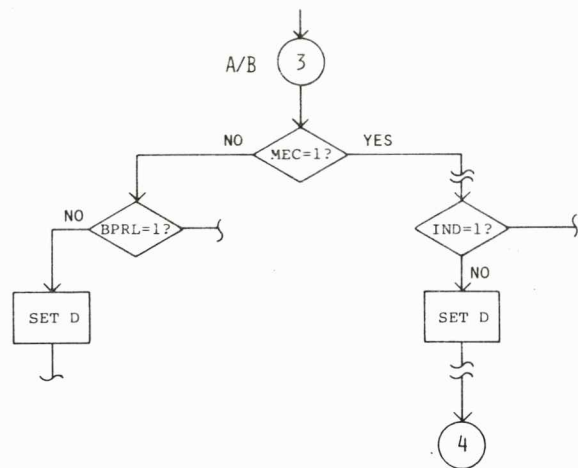
State A5 could just as easily be drawn with a single DMP ALU in the same box as the UPDOE, or in a separate box of its own, ahead of the BQ qualifier.

Such redrawing often adds a welcome measure of clarity in loops involving repeated SET IDA's and multiple qualifiers. It lets us indicate what's in D at the time a SET IDA is given after certain qualifiers have been met or failed.

FIG 18-1-4

HOW TO INTERPRET THE BPC ASM CHART, CONT.

Not all instances of the same instruction appearing twice in the same state are the result of de-minimization, however. The two SET D's in state A/B3 represent separate instances of decoding that instruction. The partial structure of the state is shown below:



Here each SET D is conditional upon a different qualifier. Because of the way the ROM is organized, an instance of an instruction being decoded represents the "AND" of selected conditions:

IF STATE 3 AND GROUP A/B AND NOT BPRL, THEN SET D
 $GP1 \cdot GP2 \cdot GP3 \leftarrow (GP0=1 \text{ for A, } 0 \text{ for B})$

The other instance of decoding SET D in that state is:

IF STATE 3 AND GROUP A/B AND NOT IND, THEN SET D

The combination:

THE CONDITION OR CONDITION, THEN SET D

does not exist as a single encoding. It is simply the "OR" (during fan-in) of the two separate "AND's" as shown above.

FIG 18-1-5

SECTION 18

Figure 18-2 illustrates the extent of the ASM chart for the BPC. It also illustrates the start-up sequence (used only at turn-on).

Figure 18-3 illustrates the instruction fetch and fan-out portion of the ASM chart. State zero is a loop for completing the instruction fetch memory cycle. (The instruction fetch process is initiated by the tail end of the previously executed section of flow charting.) It is during this loop that the flags are captured and that an interrupt is enabled. Notice that the fetched

instruction is placed not only in the I register, but also in the T register. By putting the instruction in the T register the operand of a memory reference instruction is made available to the ALU.

Once the instruction fetch is completed the contents of the R register are put into the S register. This is done in case R must be incremented or decremented. Notice that all during state zero the ALU has been kept in the ADM mode. By the time state one is reached the ALU has had time to respond to the contents of T and P to produce a

memory address (assuming that this fetched instruction was indeed a memory reference instruction). The DMP PAD, SET D, SET IDA in state one prepares that address for use. However, since it is not known until state two whether or not the contents of the T register were indeed an operand for a memory reference instruction, Start Memory is not given until an appropriate section of flow charting is reached.

By the time state two is reached the group qualifiers have been set-up. Beginning with state two the group qualifiers make a

difference. In contrast to this, the micro-instructions decoded during state zero and state one treated the group qualifiers as don't cares. Henceforth, to properly designate a box on a flow chart it is necessary not only to identify its state number, but also to identify its group.

Figure 18-4 is the groups A and B portion of the ASM chart. Groups A and B both involve a memory reference-type read memory cycle. Accordingly, the first part of this portion of the ASM chart is shared in common by both parts A and B.

OVERVIEW OF THE ASM CHART, AND DETAILS OF THE START-UP SEQUENCE

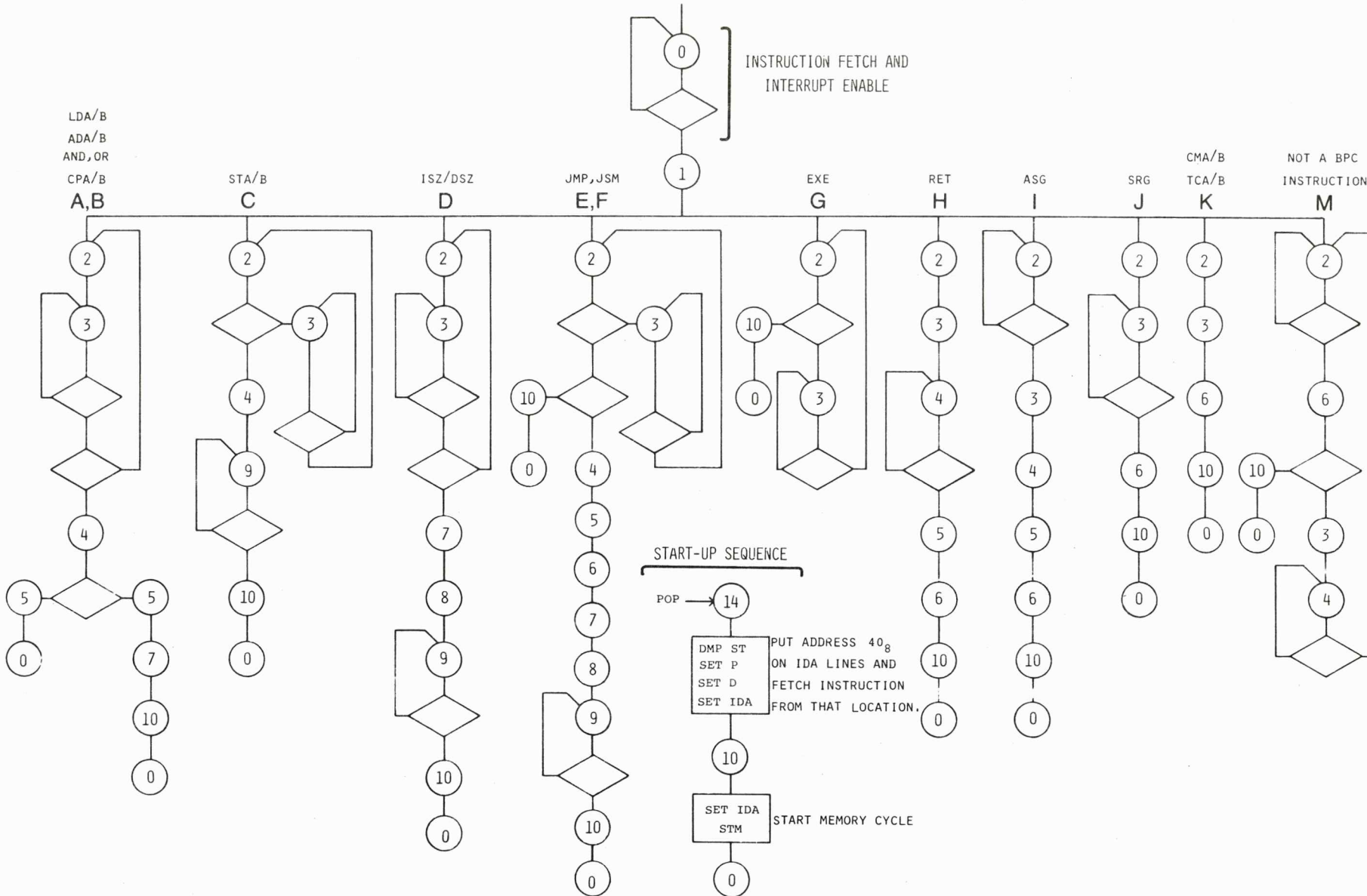


FIG 18-2

ASM CHART
OVERVIEW, START-UP
SEQUENCE
ASM CHART
INTERPRETATION (CONT)
ASM CHART
INTERPRETATION (CONT)
ASM CHART
INTERPRETATION (CONT)

PUT INSTRUCTION INTO I AND T



The disjoint sections primarily involve the following difference. Group B machine-instructions involve a possible extra increment to the value of P. That is, those machine-instructions can perform a "skip".

42

10-BIT REFERENCE
LDA M, I
INDIRECT INDICATOR

GROUP A: LDA/B
ADA/B
AND
OR
GROUP B: CPA/B

D=15 BIT ADDRESS

2
SET IDA
START MEMORY CYCLE
TO FETCH M

3
MEC=1?
NO
YES
INDIRECT BIT IS LATCHED

ADDRESS A LOCATION
WITHIN THE BPC?

NO
YES
BPRL=1?

THIS
SET D
NOT
AFFECTED
BY DRQ

DMP IDA
SET D

DMP REG
SET D
SET IDA
SEE
NOTE 1
BELOW

IND=1?
NO
YES
REFERS TO THE
REASON FOR DOING
THE PRESENT
MEMORY CYCLE

SET S

INDIRECT, NEED
NEW M. D HAS
INDIRECT ADDRESS
JUST FETCHED.
PUT IT ON IDA
BUS AS ADDRESS
FOR NEXT FETCH.

SET IDA

REFERS TO THE REASON
FOR DOING THE JUST
COMPLETED MEMORY CYCLE,
AND NOT THE VALUE OF
BIT 15 THEREBY OBTAINED.

P-ADDER HAS BEEN FORMING P+1.
UP DATE P AND PREPARE TO SEND P
OUT AS AN ADDRESS FOR NEXT
INSTRUCTION FETCH IF GROUP A

DMP PAD
SET P
SET D
SET IDA

SYNC
START GIVING SYNC

4

EXTRA TIME FOR THE SKIP MATRIX
TO SET UP AND FOR SECOND
INCREMENT OF P.

NO
YES
GROUP B?

5
SET IDA
STM

5
SET IDA
DMP ALU

7
BASED ON THE CP LINE FROM THE
ALU. INSTRUCTION DECODE
CONTROLS THE SKIP MATRIX.

NO
YES
SKP=1?

NO
YES
BQ=1?

A REG
DMP ALU
SET A

B REG
DMP ALU
SET B

DMP P

DMP PAD
P-ADDER HAS BEEN
INCREMENTING P
(ALREADY CONTAINS
P+1) TO P+2

UPDATE P REGARDLESS
AND BEGIN AGAIN
OUTPUTTING NEXT
ADDRESS FOR
INSTRUCTION FETCH

SET P
SET D
SET IDA

10
SET IDA
STM
START MEMORY CYCLE
FOR NEXT
INSTRUCTION FETCH

0

0

UPDATE OVERFLOW
AND EXTEND
UPDOE

NOTE 1: THE SET IDA PRESERVES BUS
CONVENTIONS BY MAKING THE
MEMORY FETCH AVAILABLE TO
ALL CHIPS ON THE BUS.
THE SET D IS AFFECTED BY
DRQ.

FIG 18-4

MEC, this in no way means that IND refers to bit 15 of the word just read and accompanying MEC. Quite to the contrary, IND refers to the conditions under which the just completed memory cycle was

SECTION 18 (CONTINUED)

initiated. If the memory cycle just completed was known in advance to be an application of an indirect address, then it is also known that the results of that memory cycle must also be used as an address. Likewise, IND will be false at the end of a memory cycle to fetch the final destination in an indirect chain, even though the data fetched may have a one in bit 15. This is all possible because there is a one state delay between what goes into the Indirect Bit Latch when MEC is true, and the value of the IND qualifier as asked by the ROM.

The IND qualifier is used twice in state three. During the time while memory complete is not yet met, IND indicates if the ongoing memory cycle is a fetch of a link-pointer or of the actual data. If IND is true then it's a fetch of a link-pointer. This decision is made to avoid giving a SET S until the final data is at hand. This avoids unnecessary joggling of the ALU.

When MEC is true, IND indicates whether the memory cycle just completed was a fetch of a link-pointer. That will be the case if IND is true. If IND is false the actual data is at hand. That is, the operand has finally been obtained.

No more memory cycles are necessary to complete execution of the instruction. P is replaced by its incremented value, sent out as an address and SYNC is given. However, STM is not yet given.

For group A instructions there will be no skip. Therefore, it is possible to give the second SET IDA and STM for the instruction fetch. Next, the output of the ALU is obtained and stored in the appropriate register. For group B instructions the skip condition is tested and, if necessary, the incremented value of P is incremented again. At this point the sending of a new address begins and a new instruction fetch is initiated.

An important difference exists between the 15 and 16-bit

versions with respect to the way the Indirect Bit Latch can be set. In the 15-bit version any memory cycle not done during a Bus Grant can set the Indirect Bit Latch. This is necessary to allow multi-level indirect addressing. In the 16-bit version however, the Indirect Bit Latch can be set only during an instruction fetch. It will, however, be updated at the end of every non-Bus Grant memory cycle. Those updates will necessarily clear the latch because they are not parts of instruction fetches. This limits indirect fetches to a single level.

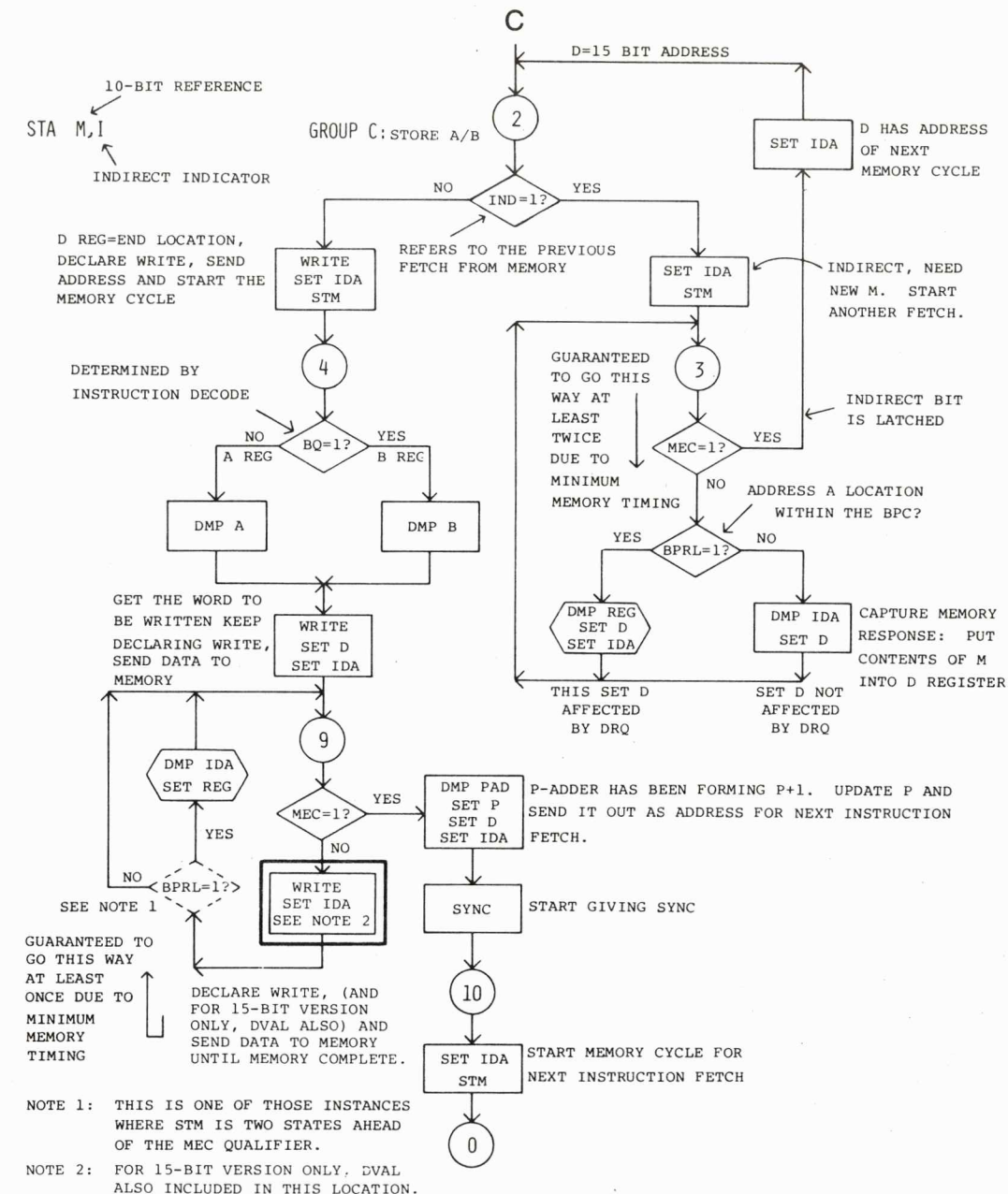
An exception to this is operation during interrupt. Recall that INT can set the latch. If INT is held long enough it can ensure that the latch is set during two consecutive memory cycles. In the 16-bit version the IOC does exactly this to ensure the necessary two levels of indirect required to perform vectored interrupt.

Figure 18-5 illustrates that segment of the ASM chart that deals with group C machine-instructions. The main characteristic of group C machine-instructions is that they involve storage into (possibly indirect) memory reference operands.

States two and three resolve the appearance of an indirect address in the same manner as for group A and B machine-instructions. If the reference was indirect the last pass through states three and two forms the final destination address. If the reference was direct then states one and two started the final memory cycle. Either way, state four dumps the register whose contents are to be written into memory.

Notice that state nine is also constructed so as to allow the destination address to be an entity within the BPC. At the conclusion of the memory cycle the value of P is updated, SYNC is given and the next instruction fetch commenced.

Figure 18-6 is a flow chart of the group D portion of the



BPC ASM CHART-ISZ AND DSZ INSTRUCTIONS

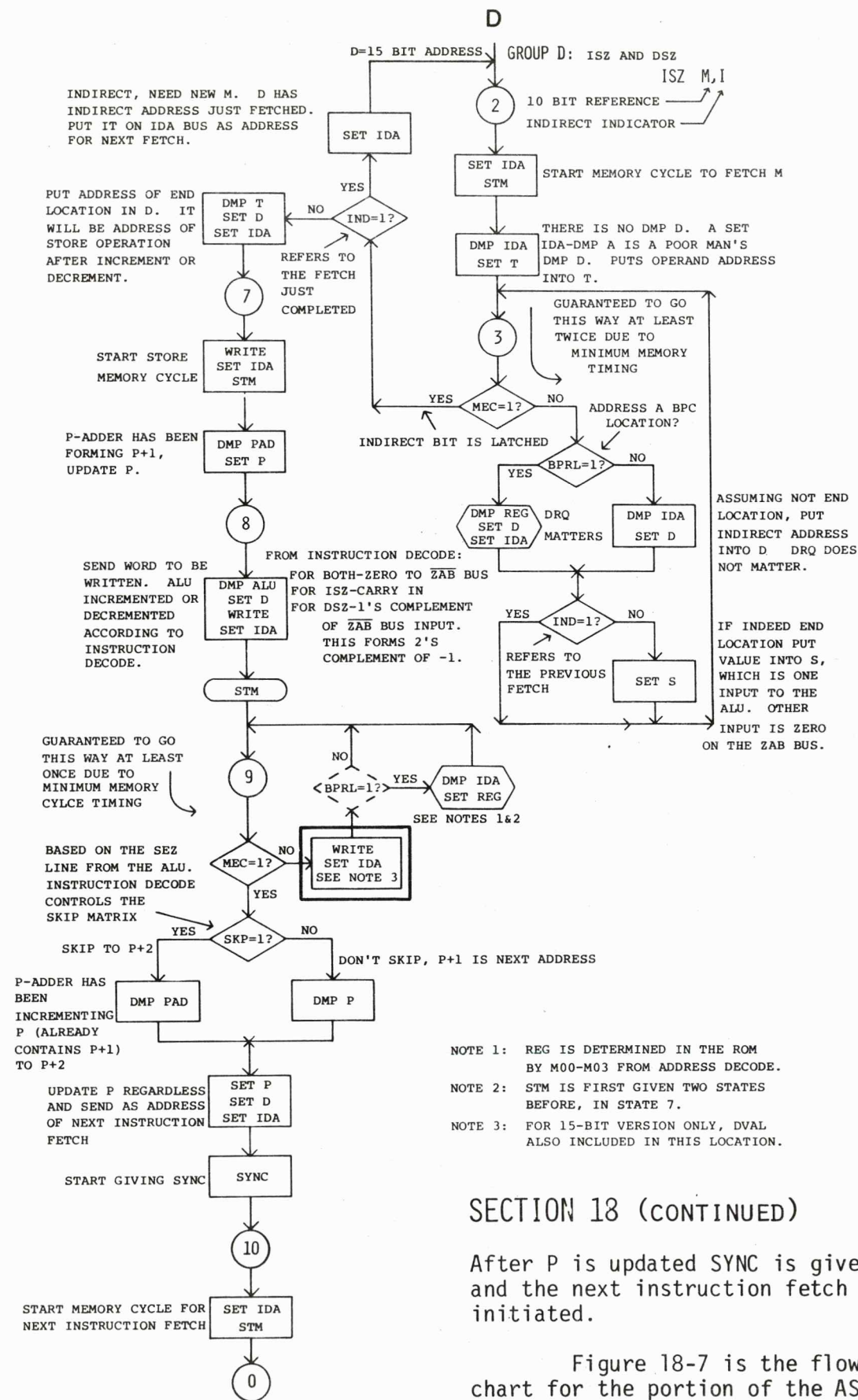


FIG 18-6

SECTION 18 (CONTINUED)

After P is updated SYNC is given and the next instruction fetch is initiated.

Figure 18-7 is the flow chart for the portion of the ASM chart corresponding to groups E and F. These groups are characterized primarily by their ability to read a possibly indirect memory

BPC ASM CHART-JUMP AND JUMP TO SUBROUTINE INSTRUCTIONS

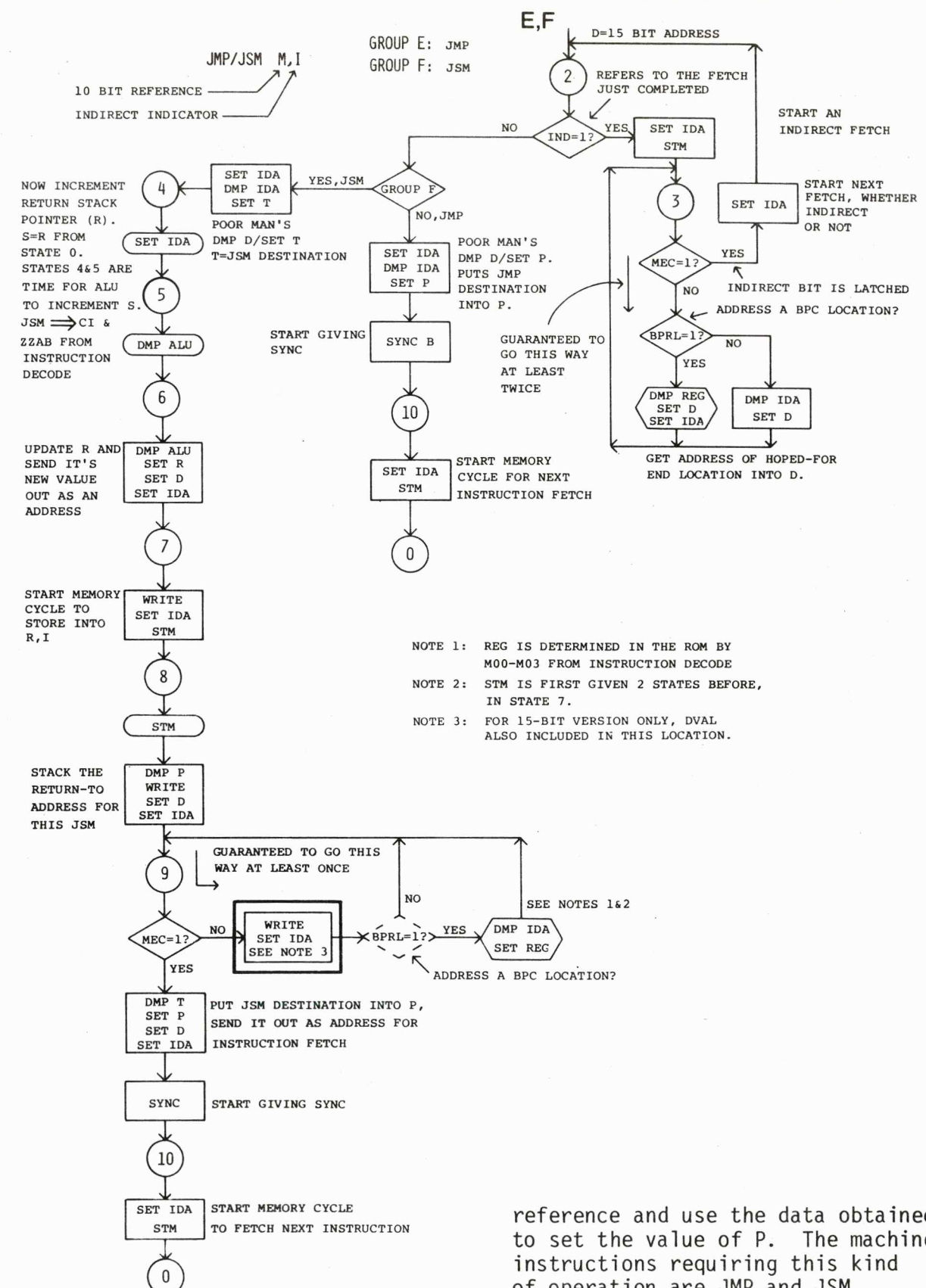


FIG 18-7

reference and use the data obtained to set the value of P. The machine-instructions requiring this kind of operation are JMP and JSM. JMP is quite simple. If the original reference was non-indirect the completed address corresponding to the operand is simply placed in P. No actual memory cycle is involved. If the original reference was in-

BPC ASM CHART-EXECUTE INSTRUCTION

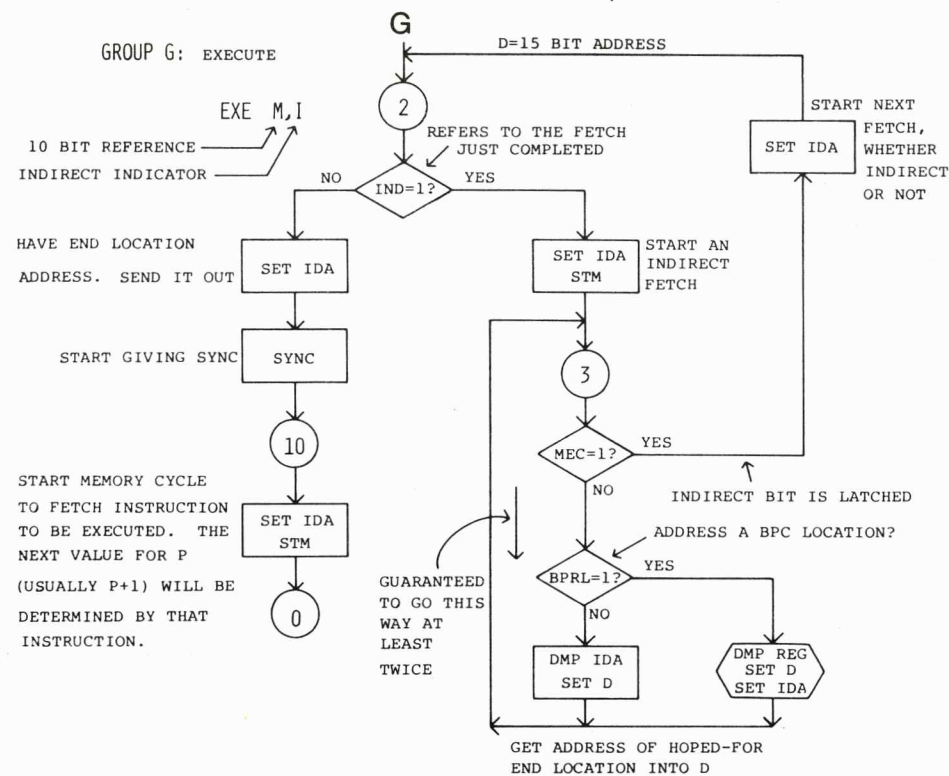


FIG 18-8

SECTION 13 (CONTINUED)

direct, as soon as the destination address is reached its contents are put into P. At that point SYNC is given and the next instruction fetch commences.

A JSM machine-instruction is similar in the manner by which it obtains the new value of P. However, there are some additional operations that must also be performed. First, the new value of P is temporarily stored in T. Then the value of the return stack pointer, R, is updated and the old value of P stored in the location now pointed to by R. Following that, the contents of T are put into P and a new instruction fetch initiated.

Figure 18-8 is a flow chart of the Execute (EXE) portion of the ASM chart. To perform the EXE machine-instruction, it is first necessary to resolve a possible indirect memory reference. Once the data value of the operand is obtained it is used as an address with which to form an instruction fetch. This is done in the same way that any instruction fetch is initiated. That

is, by a transition to state zero after sending out the address. The only difference is that the value of P is left totally alone. It will be changed according to the rules for the instruction encountered as a result of performing the Execute machine-instruction.

Figure 18-9 is a flow chart of the Return (RET) portion of the ASM chart. The execution of the RET machine-instruction requires using the R register as an indirect pointer into a stack to obtain a value of P. The value of P so obtained can be modified by a 6-bit skip field similar in nature to those of the skip machine-instructions.

States two and three utilize the contents of the R register as an address and initiate the memory cycle to obtain the raw value of P from the return stack. While waiting for MEC in state four the P-Adder is put into ADS mode in preparation for the modification of the raw value of P (by the skip field in the RET machine-instruction).

BPC ASM CHART-THE RETURN INSTRUCTION

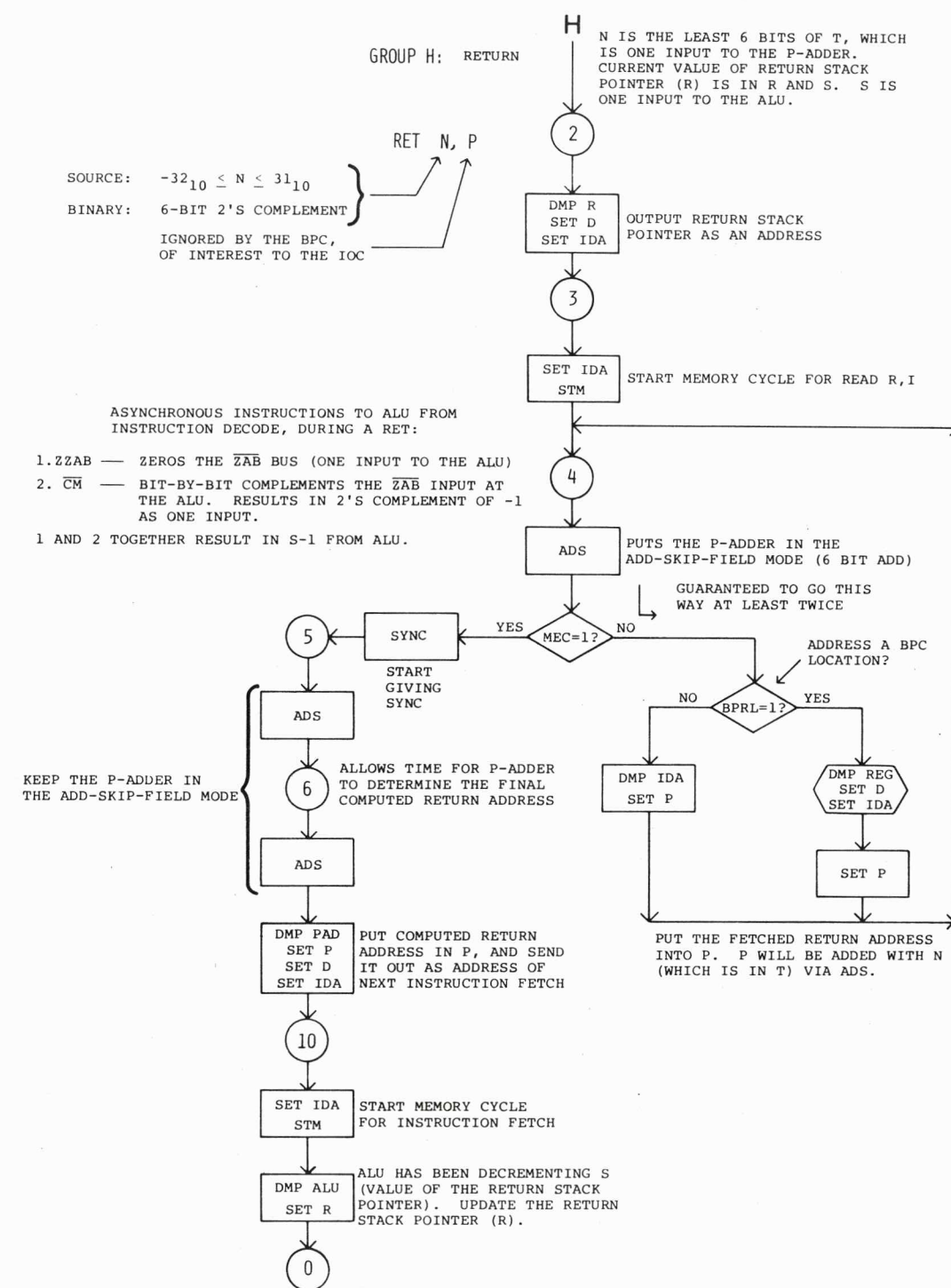


FIG 18-9

SYNC is given upon receipt of MEC, since no further memory cycles are needed.

State five allows time for the P-Adder to perform the ADS mode add operation. (The memory cycle just performed in states two, three and four actually put the value read into the P register.) In state six the modified value of P is ob-

tained and placed into the P register. It is also sent out as the address of the next instruction fetch.

Meanwhile, throughout the execution of the RET machine-instruction the ALU has been forming the decremented value of the R register. That happens in the following way. Way back in state zero the value of R was put in S. Recall that S is

BPC ASM CHART - ALTER-SKIP GROUP INSTRUCTIONS

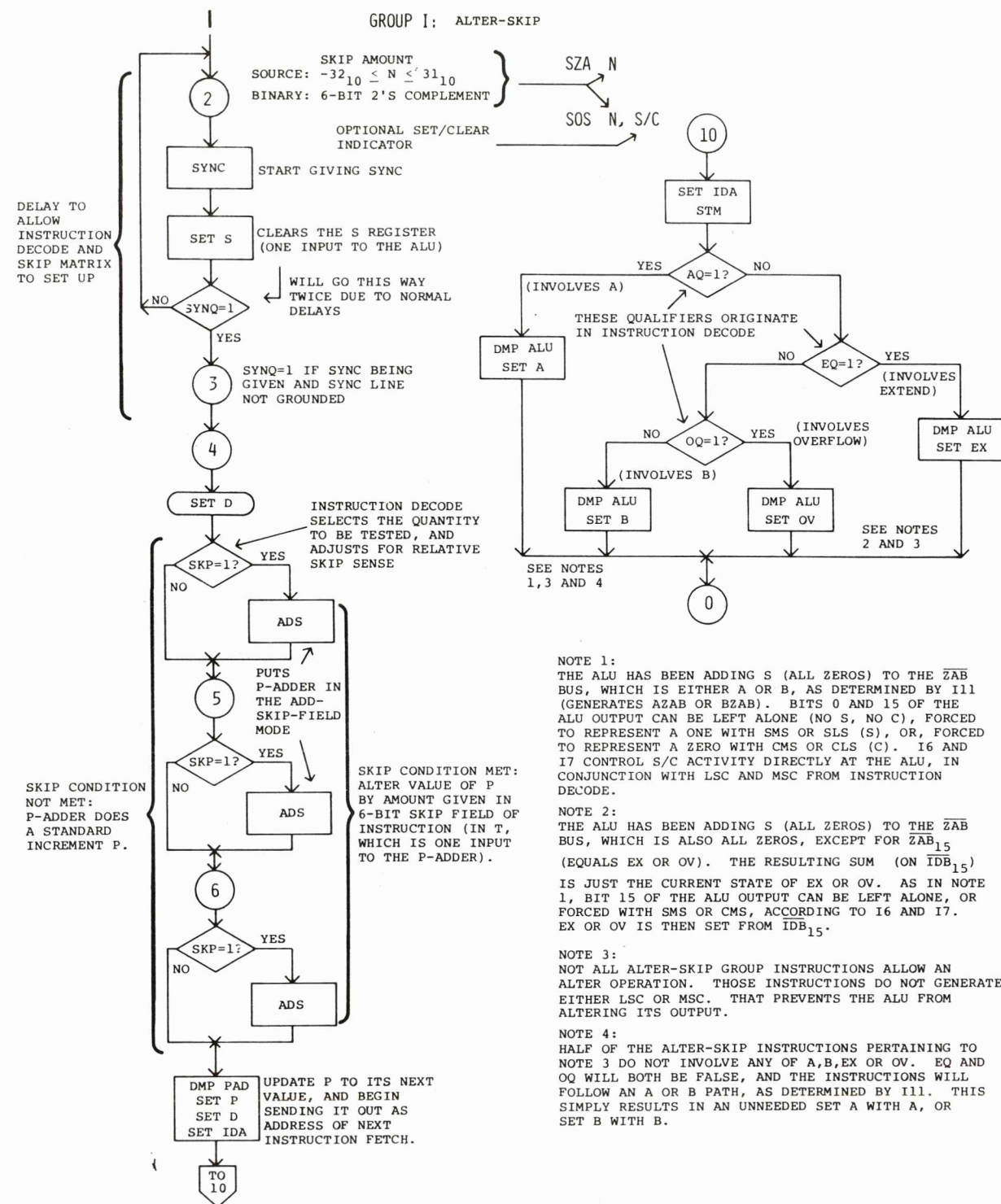


FIG 18-10

SECTION 13 (CONTINUED)

one input to the ALU. The asynchronous control instructions for this particular machine-instruction cause the other input to the ALU to represent the two's complement of -1. This is done with a ZZAB and a CM. Together, that results in the ALU forming S-1, which is really R-1.

In state ten the decremented value is placed into R.

Figure 18-10 is a flow chart of the Alter/Skip portion of the ASM chart. The execution of this group of machine-instructions requires that the following things happen. First,

BPC ASM CHART - SHIFT-ROTATE GROUP INSTRUCTIONS

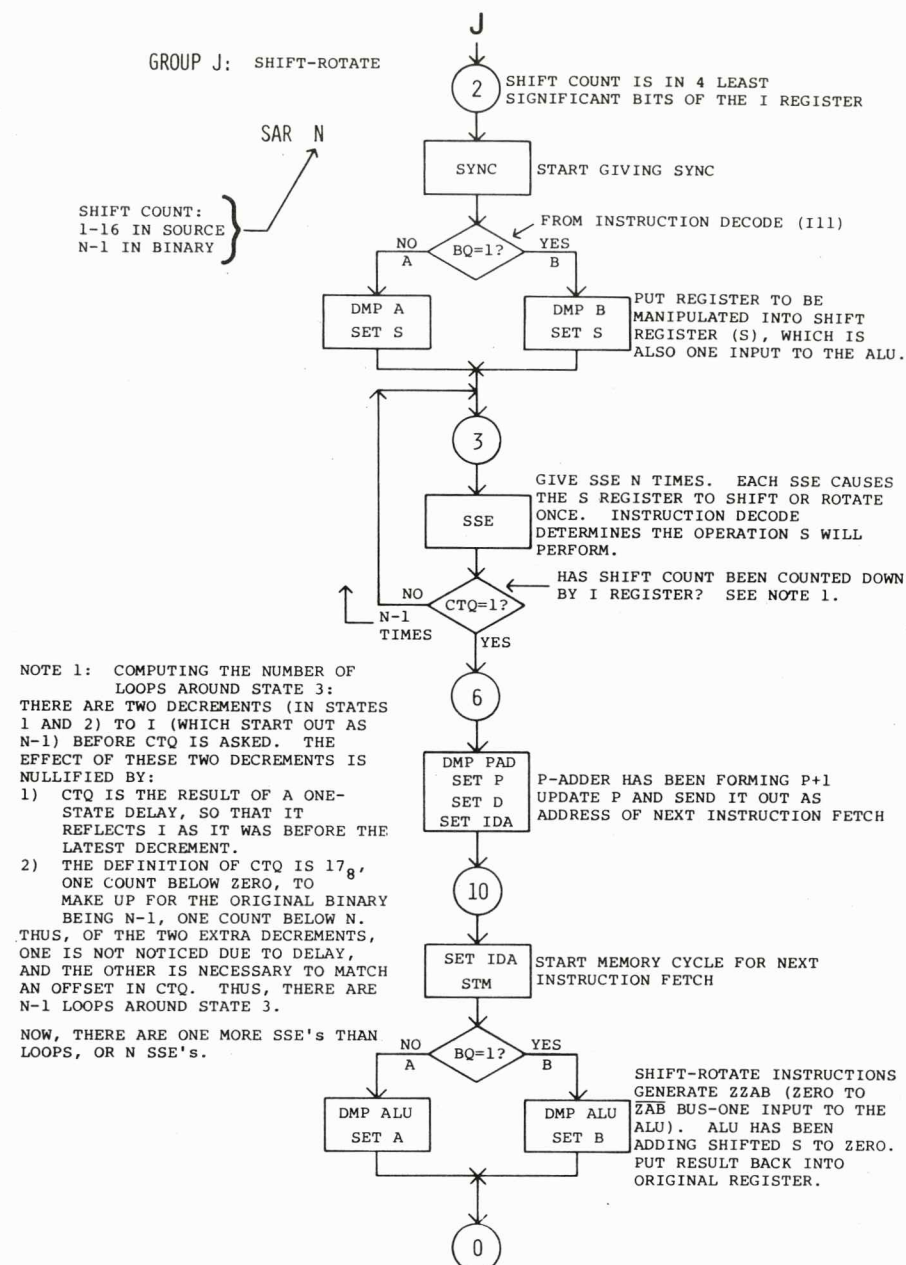


FIG 18-11

there must be a delay to allow the Skip Matrix time to set up. Next, the P register must be incremented by one if the skip condition is failed, or modified according to the skip field if the skip condition is met. Last, the latter condition, if present, must be acted upon.

States two and three provide the time required for the Skip Matrix to set up. An extra two state-times of delay is obtained by having state two give SYNC and then loop upon itself until the qualifier SYNQ is true.

States four, five and six

perform the necessary modification to the value of P. The P-Adder will perform either a standard INCP operation or a ADS mode operation, based on the output of the Skip Matrix. At the conclusion of the add operation the new value of P is actually placed in P and the new instruction fetch is initiated.

In state ten a DMP ALU, given in conjunction with the proper SET micro-instruction, performs any necessary alter operation. Notes 1 through 4 on the flow chart explain this process in detail.

BPC ASM CHART-COMPLEMENT GROUP INSTRUCTIONS

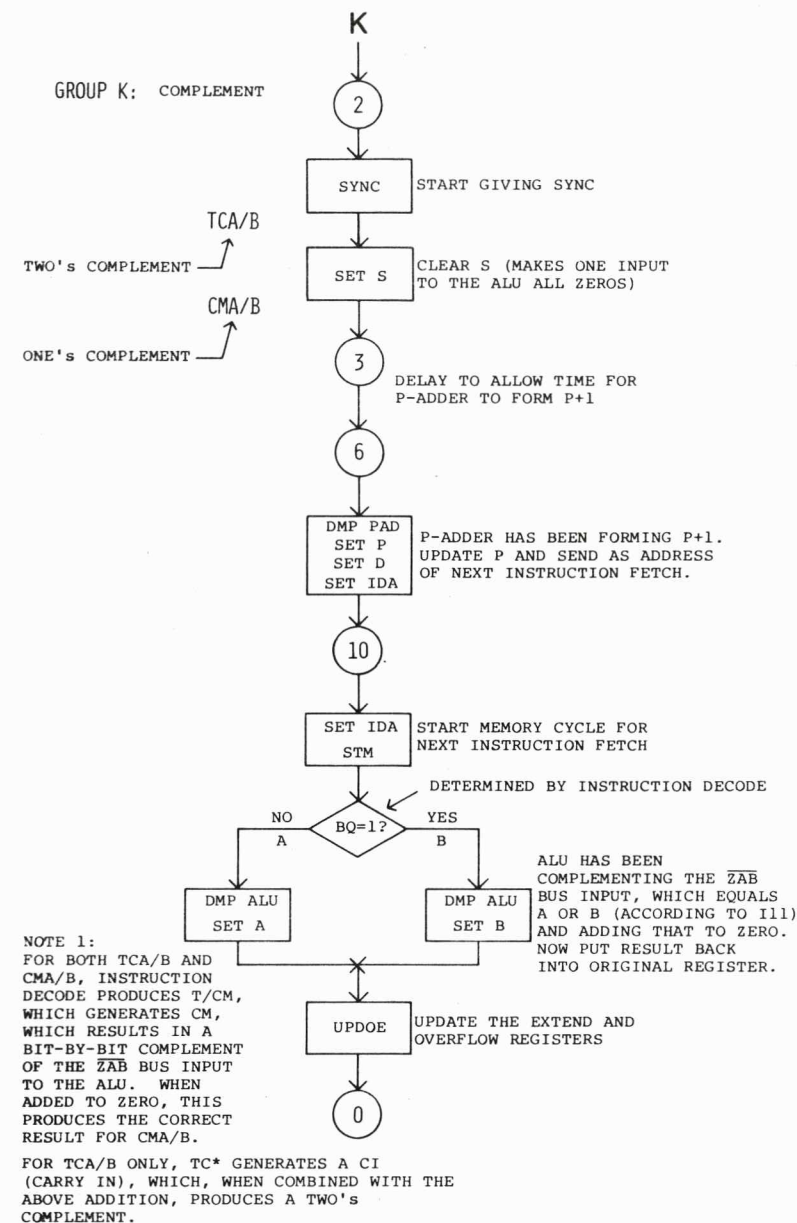


FIG 18-12

SECTION 18 (CONTINUED)

Figure 18-11 is the flow chart for the Shift-Rotate portion of the ASM chart. None of these machine-instructions perform any memory cycles, so SYNC may be given right away. In addition, these machine-instructions can shift or rotate only the A or B registers. The affected register is immediately placed into S, where it may be shifted or rotated. All of this is done in state two.

In state three the proper number of shift or rotate operations are performed upon the S register. The relationship between the count

in the least four significant bits of the I register and the qualifier CTQ is explained in notes 1 through 3 in the flow chart.

After the proper number of shift or rotate operations has been performed, P is updated and the next instruction fetch is initiated.

At the conclusion of this group of machine-instructions the modified value of S is obtained via the ALU, by adding S to zero. This output is then used to set either the A or B register, as is appropriate.

BPC ASM CHART - NON-BPC INSTRUCTION

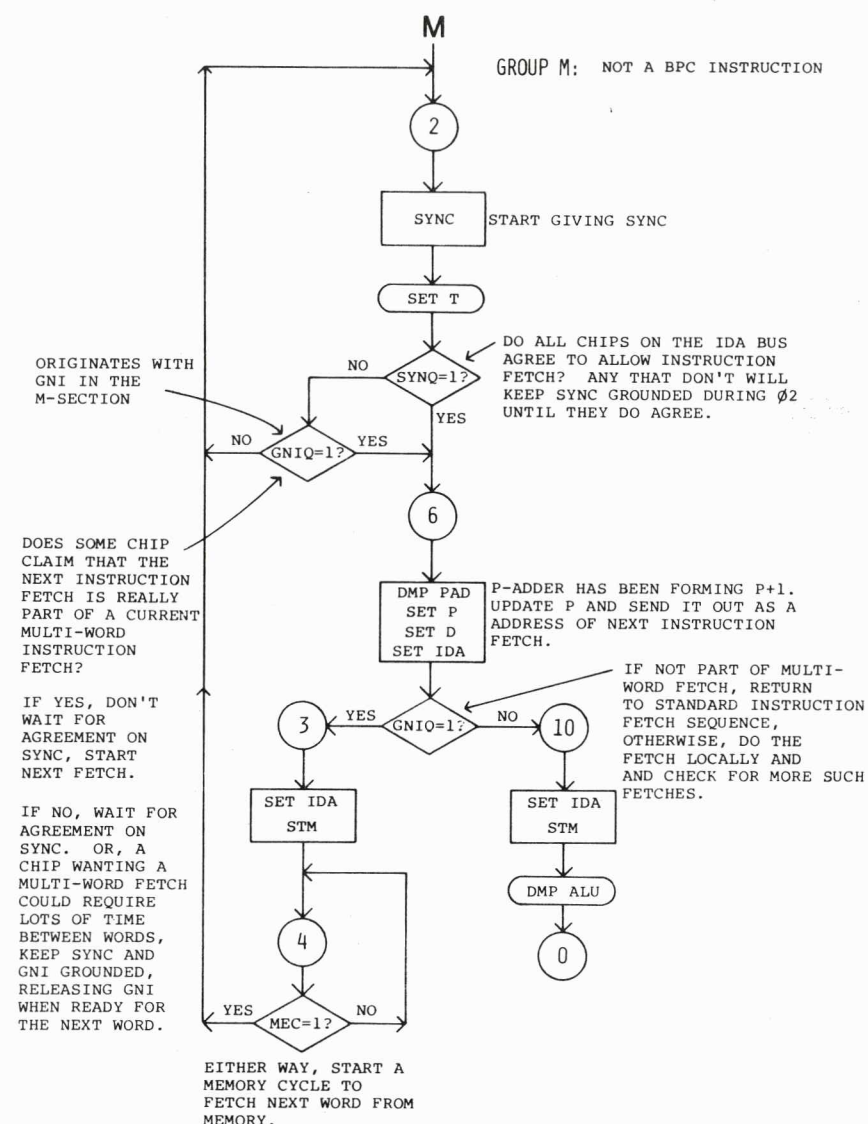


FIG 18-13

Figure 18-12 is the portion of the ASM chart dealing with Complement-type machine-instructions. Only the A and B registers can be complemented. The basis of the operation will be to send either A or B to the ALU, while the other input (the S register) is zero. Instruction decode will turn on the complements and supply, if appropriate, a carry-in. The carry-in is used to generate two's complements. No carry-in is used to generate one's complements.

Since this group of machine-instructions does not perform

any memory cycles, SYNC can be given right away. This is done in state two. Also in state two, there is a SET S with no corresponding DMP instruction. This sets the S register to all zeros.

After a delay in state three, the P register is updated in state six and the next instruction fetch is initiated.

By state ten the ALU has completed the arithmetic operation required to produce the proper complement. A DMP ALU, in conjunction with the proper SET-register micro-instruction, obtains the desired complement. Observe that the Extend and Overflow registers are updated according to the results of the addition just performed.

Figure 18-13 shows that portion of the BPC's ASM chart that is accessed whenever the fetched instruction is a non-BPC instruction. This section of the ASM chart has two purposes. First, it controls the BPC's response to SYNC, when SYNC is controlled by another chip. Second, it implements the ability of some chip on the Bus to employ a multiple-word instruction fetch. This later capability employs the signal Get Next Instruction (GNI). GNI is normally pulled up so that the qualifier GNIQ will normally be false.

Immediately upon reaching state two the BPC gives SYNC. However, this does not mean that the actual SYNC line goes true. Since the fetched machine-instruction belongs to another chip, it is quite possible that that chip is keeping SYNC grounded for a much longer period of time than it has taken the BPC to reach state two. The SYNC line will go high only after *all* chips have agreed to let it do so, *and* the BPC has given SYNC.

A multiple-word instruction fetch would work as follows. Upon reaching state two both SYNQ and GNIQ would be false. The entity requiring the multiple-word instruction fetch would cycle GNI each time it wished the next word. SYNC, however, would remain grounded. When GNIQ goes true, P is incremented and state three performs an instruction fetch.

NON-BPC
INSTRUCTION

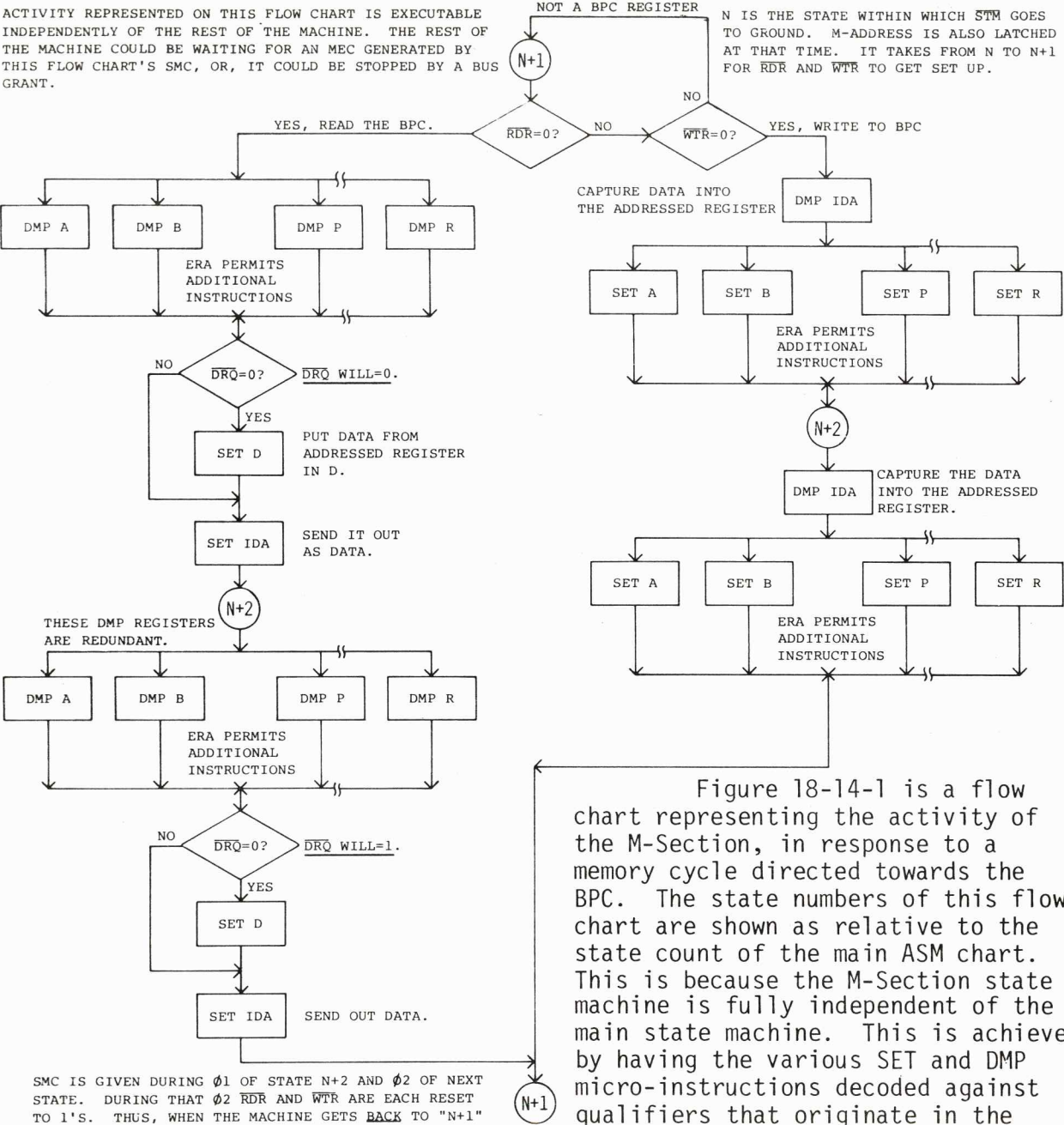
COMPLEMENT
EXECUTE

SHIFT-ROTATE

ALTER/SKIP

BPC FLOWCHART FOR THE M-SECTION STATE MACHINE

ACTIVITY REPRESENTED ON THIS FLOW CHART IS EXECUTABLE INDEPENDENTLY OF THE REST OF THE MACHINE. THE REST OF THE MACHINE COULD BE WAITING FOR AN MEC GENERATED BY THIS FLOW CHART'S SMC, OR, IT COULD BE STOPPED BY A BUS GRANT.



SMC IS GIVEN DURING $\phi 1$ OF STATE N+2 AND $\phi 2$ OF NEXT STATE. DURING THAT $\phi 2$ RDR AND WTR ARE EACH RESET TO 1'S. THUS, WHEN THE MACHINE GETS BACK TO "N+1" ON THIS DRAWING, IT IDLES UNTIL ANOTHER BPC-REGISTER MEMORY CYCLE IS STARTED.

FIG 18-14-1

SECTION 18 (CONTINUED)

At the conclusion of that instruction fetch the state machine returns to state two to repeat the entire process.

To return to the standard instruction fetch sequence, or to use it in the first place, GNI is left false when SYNC is given. This allows the BPC to merely increment P and begin the next normal instruction fetch.

Figure 18-14-1 is a flow chart representing the activity of the M-Section, in response to a memory cycle directed towards the BPC. The state numbers of this flow chart are shown as relative to the state count of the main ASM chart. This is because the M-Section state machine is fully independent of the main state machine. This is achieved by having the various SET and DMP micro-instructions decoded against qualifiers that originate in the M-Section. The group qualifiers and the regular state count do not affect the decoding of these micro-instructions.

The purpose of the M-Section state machine is to respond to read and write memory cycles directed toward entities that reside within the BPC. It does this even though the main ASM chart may have been the originator of the memory cycle.

Figure 18-14-2 is a tabular listing of ERA addresses and their associated entities within the BPC. ERA mode addressing is implemented by a combination of attributes of Address Decode and of the M-Section.

BPC ERA ADDRESSING

IDA ADDRESS (OCTAL)	READ/WRITE	RESULTING u-INSTRUCTION (S)
(NOTE 1)		
40	R	DMP PAD (,INC P)
41	R	DMP PAD, ADS
42	R	DMP PAD, ADM
42	R	DMP PAD (,INC P)
44	R	DMP EX
44	W	SET EX
45	R	DMP OV
45	W	SET OV
46	R	DMP T
46	W	SET T
47	W	SET D
51	R	DMP ALU (NOTE 3)
52	W	SET S
53	R	DMP ST
53	W	SET I
54	R	DMP A
54	W	SET A
55	R	DMP B
55	W	SET B
56	R	DMP P
56	W	SET P
57	R	DMP R
57	W	SET R

NOTES

- IN THE ERA MODE, THE BPC REGISTER ADDRESS DETECTOR RESPONDS TO IDA BUS ADDRESSES, AS SHOWN. HOWEVER, ONLY THE FOUR LEAST SIGNIFICANT BITS OF THAT ADDRESS ARE USED AS QUALIFIERS IN THE ROM. HENCE, WHILE 57_8 IS DECODED (FOR R), THE SET R AND DMP R IN THE ROM RESPOND TO 17_8 ON THE M-ADDRESS LINES.
- INC P (P-ADDER OUTPUT = P+1) IS THE NOR OF ADM AND ADS. IN GENERAL, THE RESULT OF A DUM PAD IS DIFFICULT TO PREDICT, AS IT DEPENDS UPON WHERE IN ITS FLOW CHART THE BPC IS STOPPED.
- IN GENERAL, THE RESULT OF A DMP ALU IS DIFFICULT TO PREDICT. IT DEPENDS UPON WHAT IS IN THE I, A, AND B REGISTERS.

FIG 18-14-2

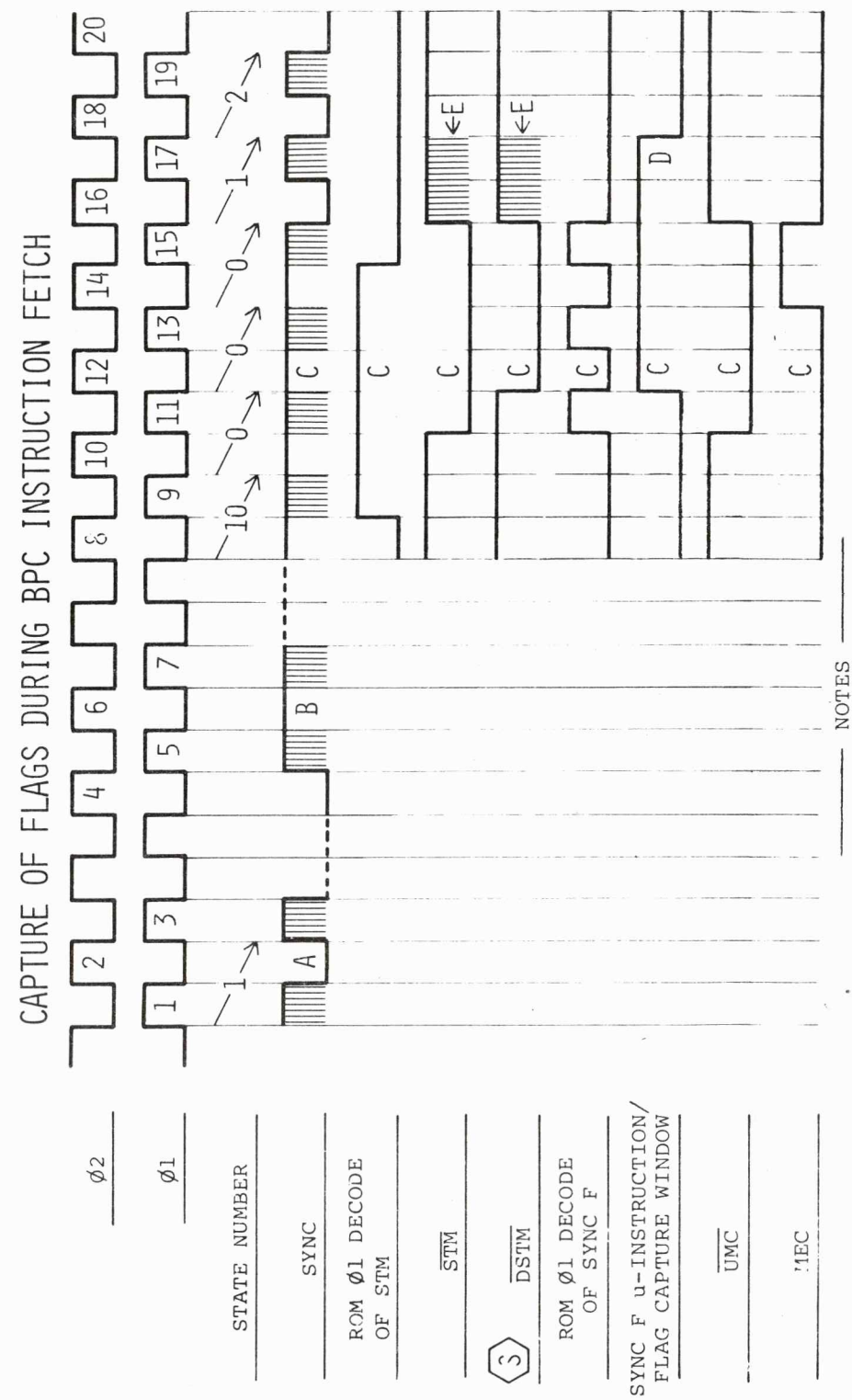
- CONVENTIONS USED IN THE WAVEFORMS**
- OR

TRANSITION UP OR TRANSITION DOWN CAN OCCUR ANYTIME WITHIN THE INDICATED INTERVAL. USED TO INDICATE TIME-WINDOWS WITHIN WHICH EXTERNALLY ORIGINATED EVENTS CAN HAPPEN. REPRESENTS IDEALIZED LOGICAL ACTIVITY; RISE TIMES AND DELAYS ARE TAKEN INTO CONSIDERATION ONLY IN A GENERAL WAY.
 - REPRESENTS THE SET-UP TIME OF A SIGNAL BEING DRIVEN.
 - REPRESENTS A LINE THAT IS EITHER UNDEFINED OR A DON'T CARE.
 - REPRESENTS A LINE THAT IS ACTIVELY PULLED-UP.
 - CAPITAL LETTERS FROM THE START OF THE ALPHABET REPRESENT EXPLANATORY NOTES. LETTERS FROM THE END OF ALPHABET ARE SOMETIMES USED TO DENOTE DIFFERENT STATES (IN THE ROM).
 - NUMERALS IN THE Ø2-Ø1 WAVEFORMS ARE STRICTLY FOR REFERENCE WITHIN ANY PARTICULAR SET OF WAVEFORMS, AND HAVE NO SIGNIFICANCE OUTSIDE THAT SET.
 - IN GENERAL, THE WAVEFORMS ARE QUITE IDEALIZED. THEY EXPLAIN THE LOGICAL RELATIONSHIPS BETWEEN SIGNALS, BUT ACTUAL DELAYS, RISE TIMES, SIGNAL LEVELS AND THRESHOLDS ARE NOT INDICATED.
 - F

DENOTES THAT A SIGNAL OCCURS IN THE 15-BIT VERSION ONLY. DELETE THE SIGNAL FOR THE 16-BIT VERSION.
 - S

DENOTES THAT A SIGNAL OCCURS IN THE 16-BIT VERSION ONLY. DELETE THE SIGNAL FOR THE 15-BIT VERSION.

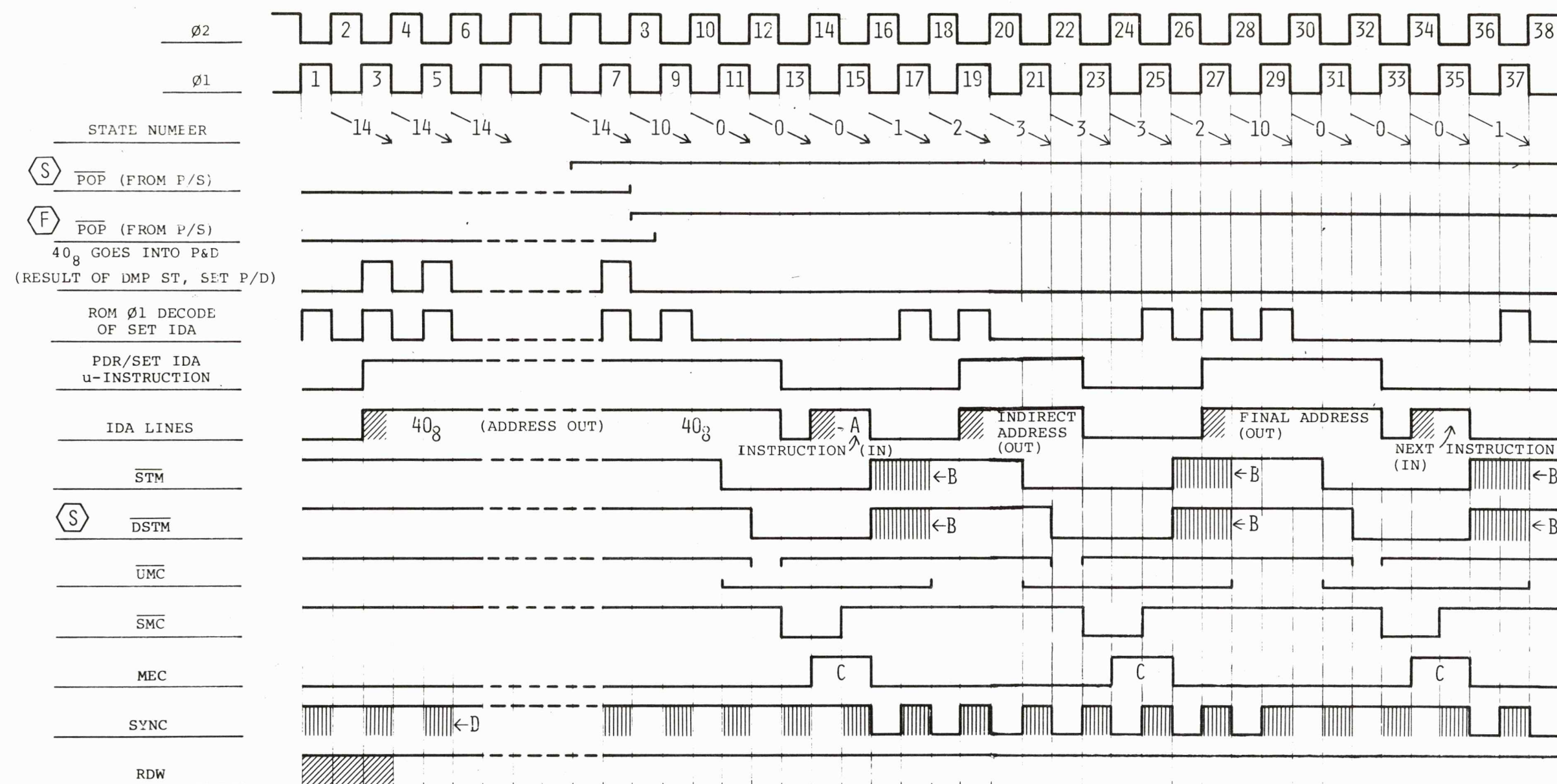
FIG 19-1



- SYNC IS FALSE AT THE END OF THE PREVIOUS INSTRUCTION FETCH.
- DURING THE EXECUTION OF THE PREVIOUS INSTRUCTION, SYNC GOES TRUE ON OR BEFORE STATE 10.
- CAN BE IN STATE 0 FOR 2, 3, OR 4 STATES. TYPICALLY IT IS 3. THE EXACT NUMBER DEPENDS UPON WHEN STM WAS GIVEN, AND WHETHER OR NOT THE FETCH IS FROM A BPC REGISTER.
- AT VERY LEAST, FLAGS MUST BE TRUE DURING THIS Ø2.
- ACTIVE PULL-UP DURING MECD.

FIG 19-2

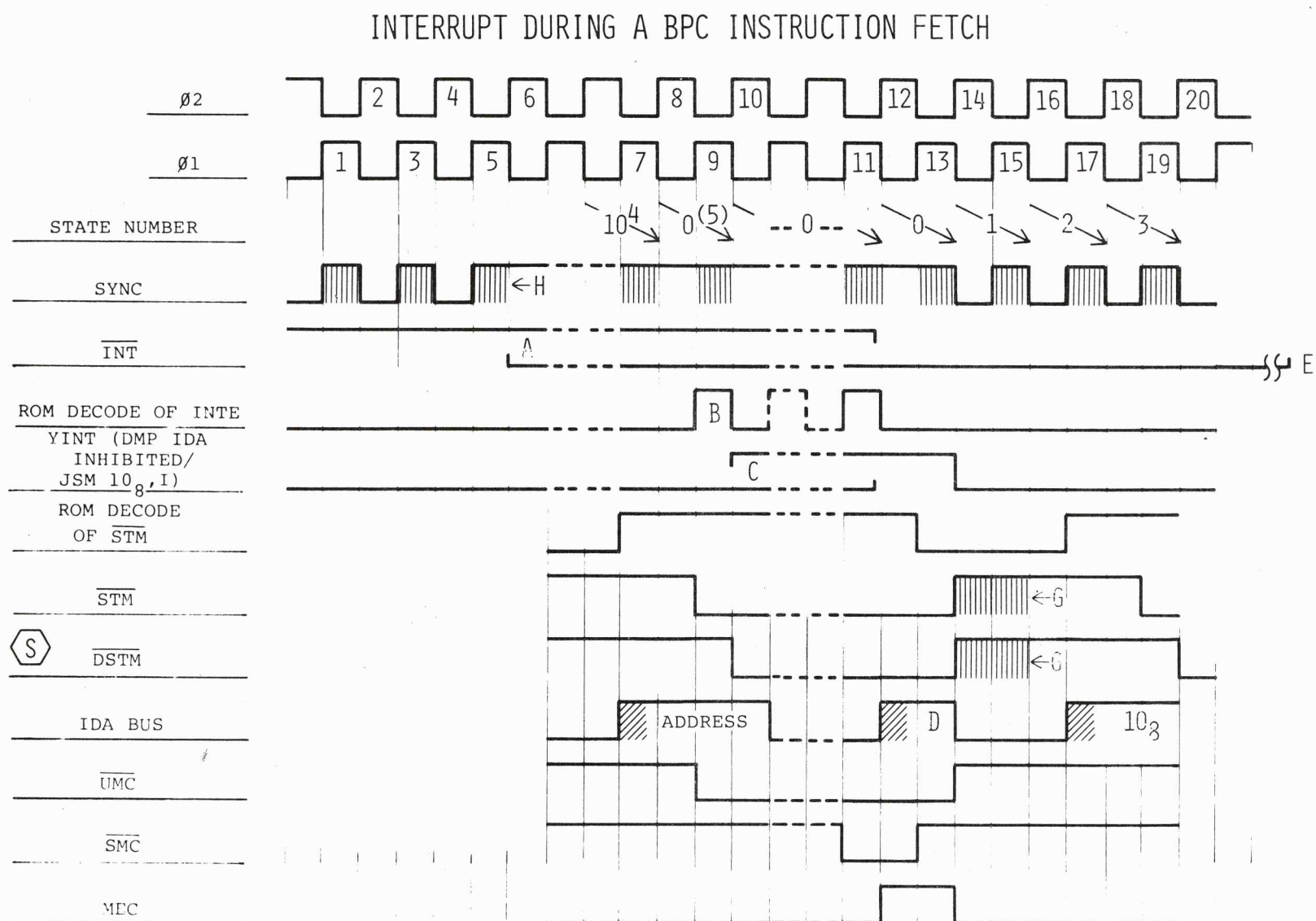
BPC START-UP AND FIRST INSTRUCTION FETCH SEQUENCE



NOTES

- IDA EQUALS THE INSTRUCTION FETCHED FROM LOCATION 40₈. IN THIS EXAMPLE WE ASSUME THAT IT IS JMP 41₈, I.
- ACTIVE PULL-UP ON \overline{STM} AND \overline{DSTM} DURING MECD.
- MEC RESETS THE LATCHED ROM OUTPUT FOR SYNC.
- POP FORCES AN INITIAL SYNC. SYNC IS ALWAYS PRE-CHARGED ON 01.
- DOTTED LINES REPRESENT AN INDEFINITE DELAY.

FIG 19-3



NOTES

- A. $\overline{\text{INT}}$ MUST NOT BE PULLED BEFORE SYNC IS GIVEN.
- B. IF THE PREVIOUS INSTRUCTION WAS A GROUP A INSTRUCTION, ONE LESS STATE-TIME IS SPENT IN STATE 0, AND THIS PULSE DOES NOT OCCUR. ALSO, THE STATE NUMBERS CHANGE AS INDICATED IN THE PARENTHESES.
- C. DEPENDS UPON $\overline{\text{INT}}$. IF NOTE B APPLIES, YINT'S EARLIEST TRANSITION IS A STATE LATER THAN CLOCKTIME #10.
- D. FETCHED INSTRUCTION THAT IS ABORTED AND REPLACED INTERNALLY BY JSM 10₈, I.
- E. $\overline{\text{INT}}$ MUST BE RELEASED PRIOR TO CLOCKTIME #23, UNLESS IT IS INTENDED TO CONTINUE HOLDING $\overline{\text{INT}}$ LOW AS A MEANS TO ENSURE ANOTHER LEVEL OF INDIRECT ACCESS IN FORMING THE INTERRUPT VECTOR. THE 15-BIT IOC HOLDS $\overline{\text{INT}}$ LOW UNTIL THE START OF CLOCKTIME #35, WHICH CREATES A RACE CONDITION WITH AN AMBIGUOUS OUTCOME. IN GENERAL THE EXACT CLOCKTIME # INVOLVED IS DIFFICULT TO SPECIFY, SINCE IT HAS TO DO WITH HOW LONG IT TAKES THE INTERRUPTING AGENCY TO RESPOND WITH THE CONTENTS OF REGISTER 10₈. THE KEY TO UNDERSTANDING THIS IS TO REALIZE THAT KEEPING $\overline{\text{INT}}$ LOW CAN SET THE INDIRECT BIT LATCH IN THE M-SECTION, CAUSING ANOTHER LEVEL OF INDIRECT ACCESS.
- F. 15-BIT VERSION OF THE BPC DOES NOT ALLOW AN INSTRUCTION FETCH FROM A BPC REGISTER TO BE INTERRUPTED.
- G. ACTIVE PULL-UP DURING MECD.
- H. BPC PRE-CHARGES SYNC EVERY 01.

FIG 19-4

READ A BPC REGISTER

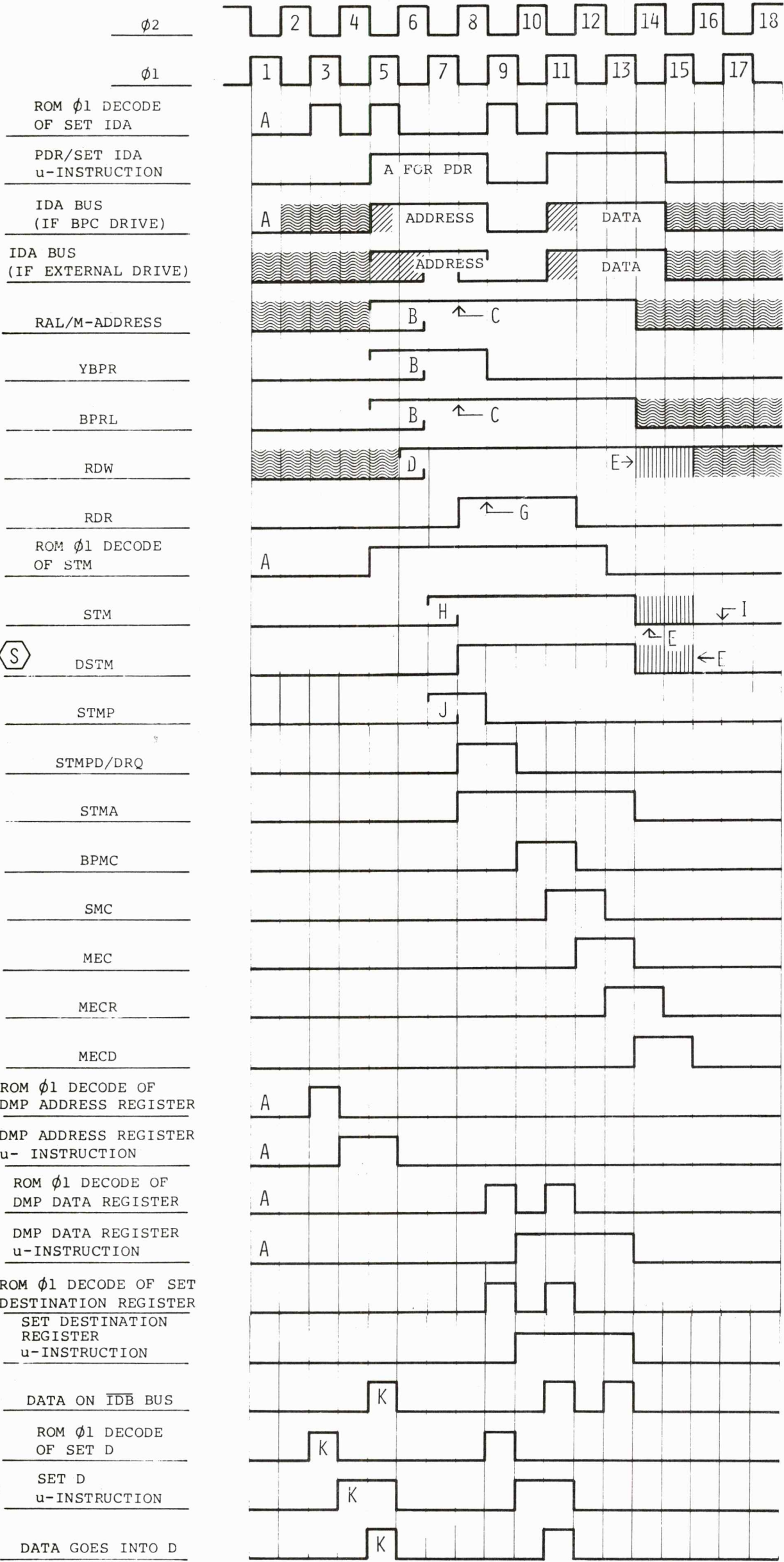


FIG 19-5

- NOTES
- A. PRESENT ONLY IF THE BPC IS THE ORIGINATOR OF THE MEMORY CYCLE.
 - B. DEPENDS UPON WHEN THE ADDRESS ON THE IDA BUS STABILIZES.
 - C. LATCHED WHEN STMA IS TRUE.
 - D. IF THIS READ MEMORY CYCLE IMMEDIATELY FOLLOWS A PREVIOUS WRITE CYCLE, THE START OF THIS $\phi 2$ IS WHEN RDW WILL GO HIGH.
 - E. ACTIVE PULL-UP OF RDW, STM AND DSTM DURING MECD.
 - F. THIS NOTE HAS BEEN DELETED.
 - G. LATCHED WHEN STMP IS FALSE.
 - H. TRANSITIONS AT THE START OF $\phi 1$ IF THE BPC IS THE ORIGINATOR. AN EXTERNAL AGENCY HAS UNTIL PRIOR TO $\phi 2$.
 - I. EARLIEST NEXT STM.
 - J. FOLLOWS STM.
 - K. PRESENT ONLY IF THE BPC IS THE ORIGINATOR OF THE MEMORY CYCLE. PART OF SENDING OUT THE ADDRESS AND THE DATA ON THE IDA BUS TO PRESERVE THE BUS CONVENTIONS.

BPC READS MEMORY (TWO CONSECUTIVE FASTEST CYCLES SHOWN)

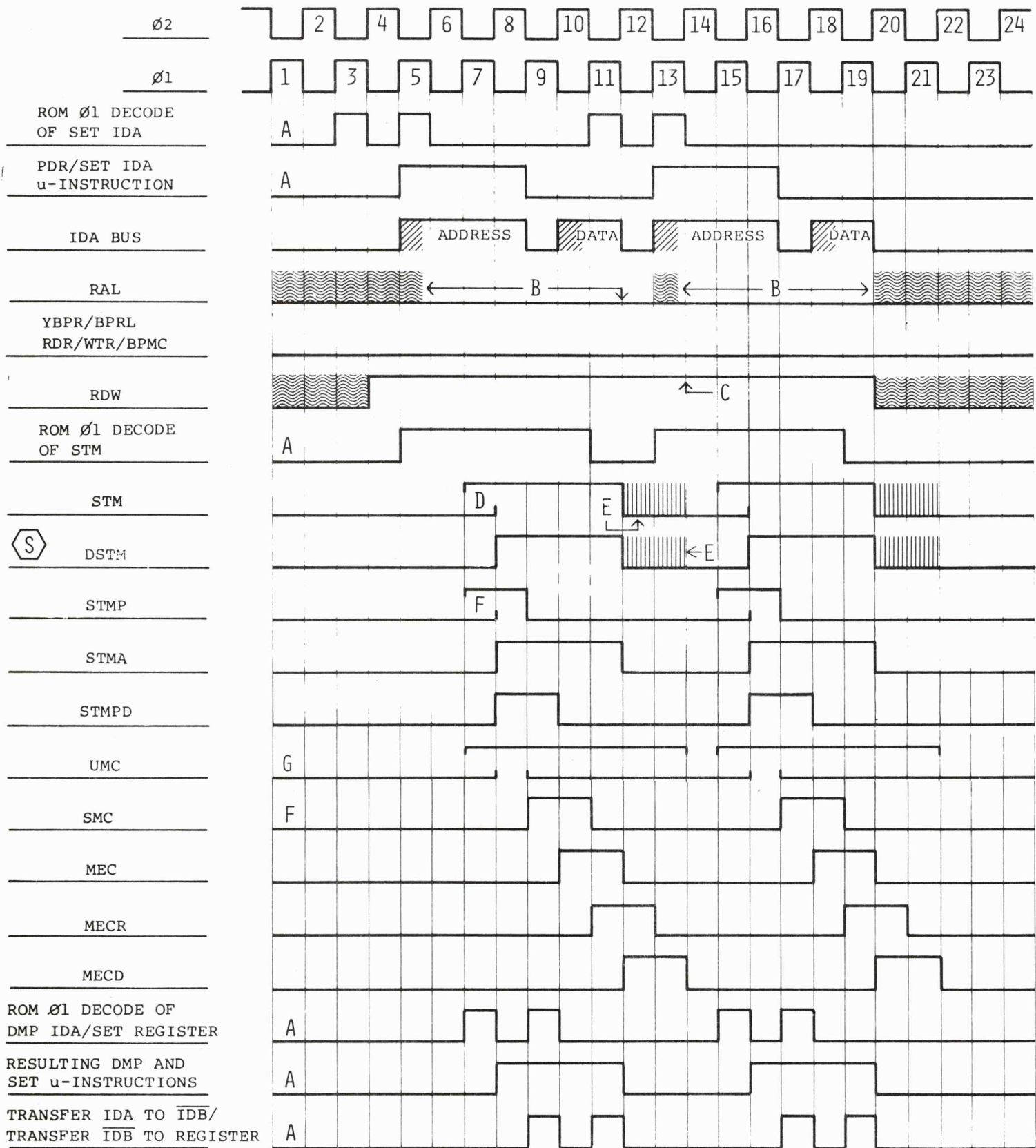


FIG 19-6

GENERALIZED 4-STATE BPC READ MEMORY CYCLE (WITH IMPLICIT HANDSHAKE BASED ON UMC AND SMC)

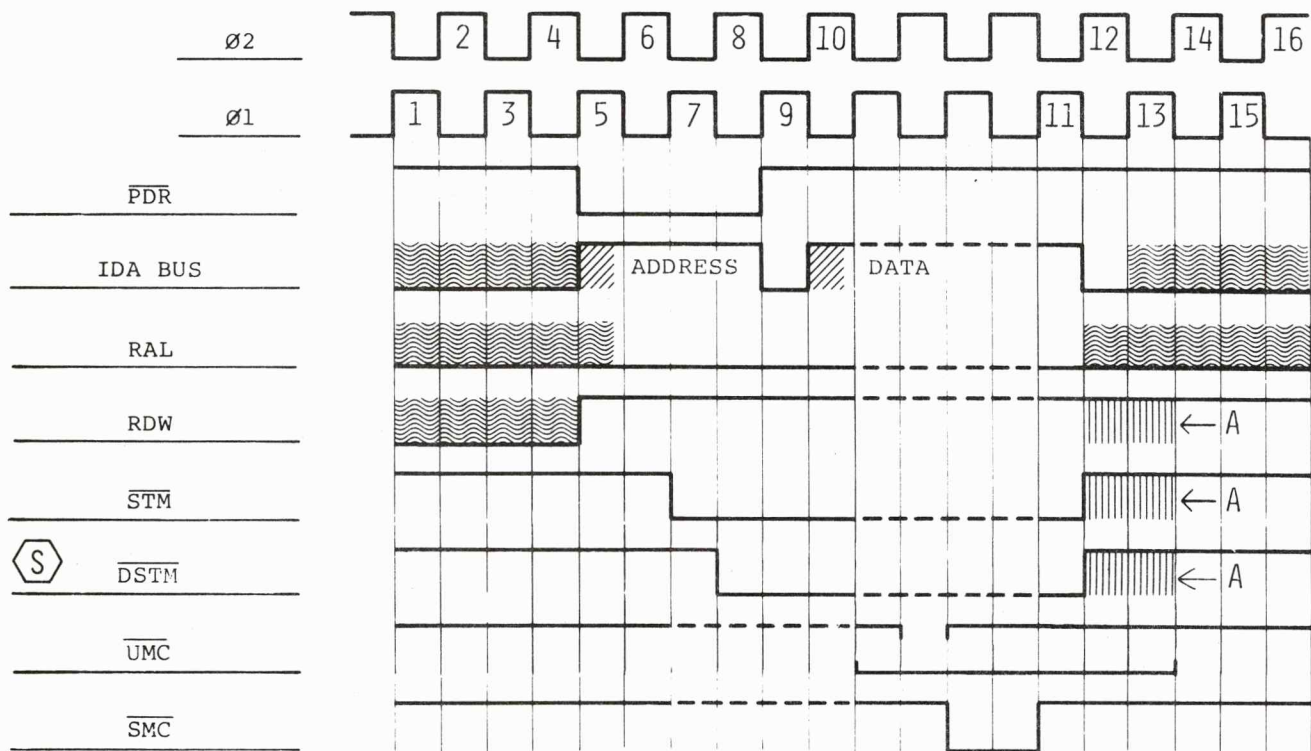


FIG 19-7

READ A BPC REGISTER

BPC READS MEMORY

4-STATE
READ MEMORY CYCLE

WRITE TO A BPC REGISTER

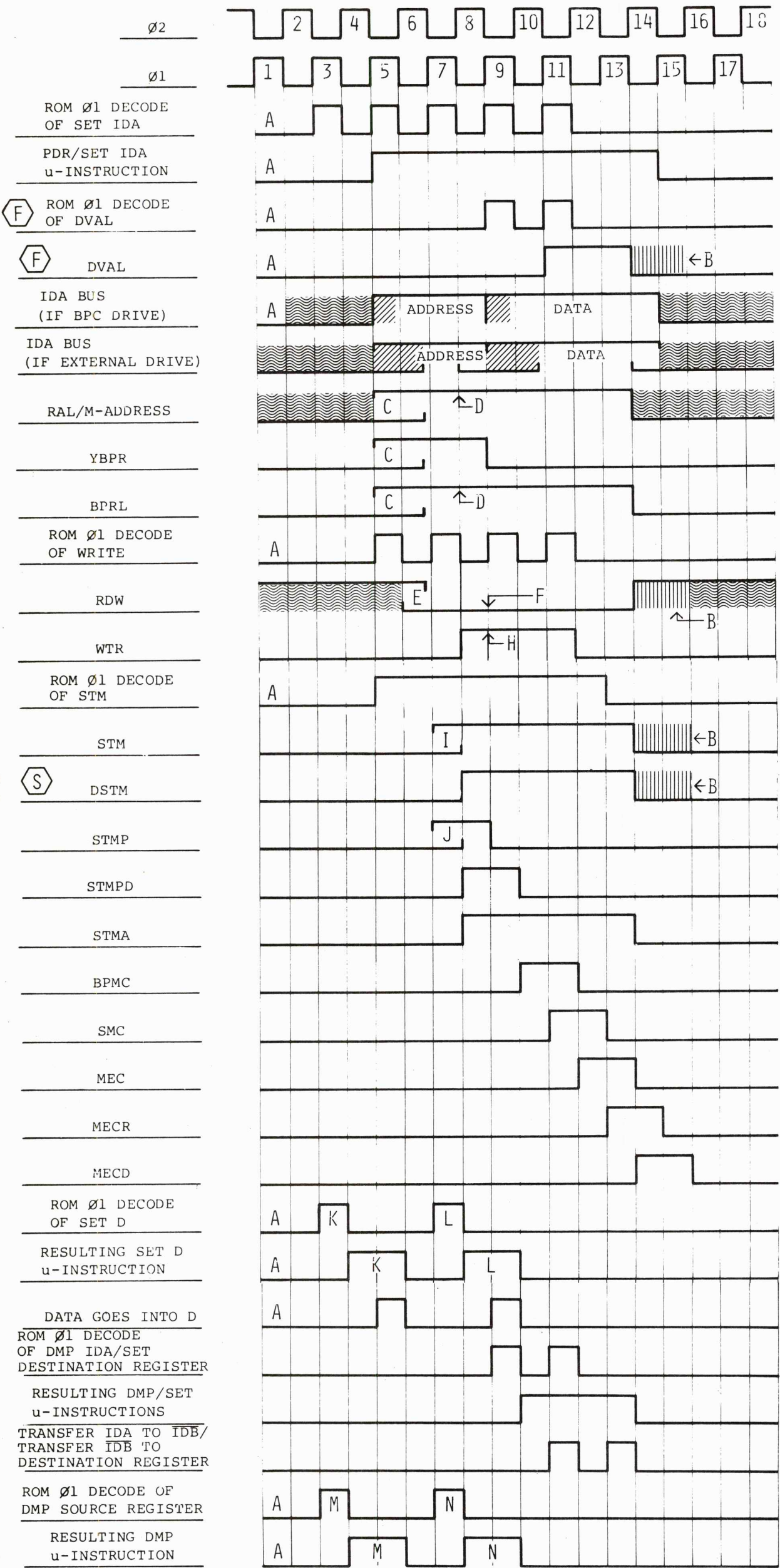


FIG 19-8

- NOTES
- A. PRESENT ONLY IF THE BPC IS THE ORIGINATOR OF THE MEMORY CYCLE.
 - B. ACTIVE PULL-UP OF DVAL, DSTM, RDW AND STM DURING MECD.
 - C. DEPENDS UPON WHEN THE ADDRESS ON THE IDA BUS STABILIZES.
 - D. LATCHED WHEN STMA GOES TRUE.
 - E. IF THE BPC IS THE ORIGINATOR, RDW WILL TRANSITION AT THE START OF 02. AN EXTERNAL AGENCY HAS UNTIL THE END OF 02.
 - F. EARLIEST RELEASE OF RDW IF AN EXTERNAL AGENCY WERE THE ORIGINATOR OF THE MEMORY CYCLE.
 - G. THIS NOTE HAS BEEN DELETED.
 - H. LATCHED WHEN STMP GOES FALSE.
 - I. TRANSITIONS AT THE START OF 01 IF THE BPC IS THE ORIGINATOR. AN EXTERNAL AGENCY HAS UNTIL PRIOR TO 02.
 - J. FOLLOWS STM.
 - K. FOR THE ADDRESS.
 - L. FOR THE DATA.
 - M. FOR THE ADDRESS. SEE NOTE N.
 - N. FOR THE DATA. ADDRESS AND DATA TYPICALLY INVOLVE DIFFERENT REGISTERS.

WRITE MEMORY (TWO CONSECUTIVE FASTEST CYCLES SHOWN)

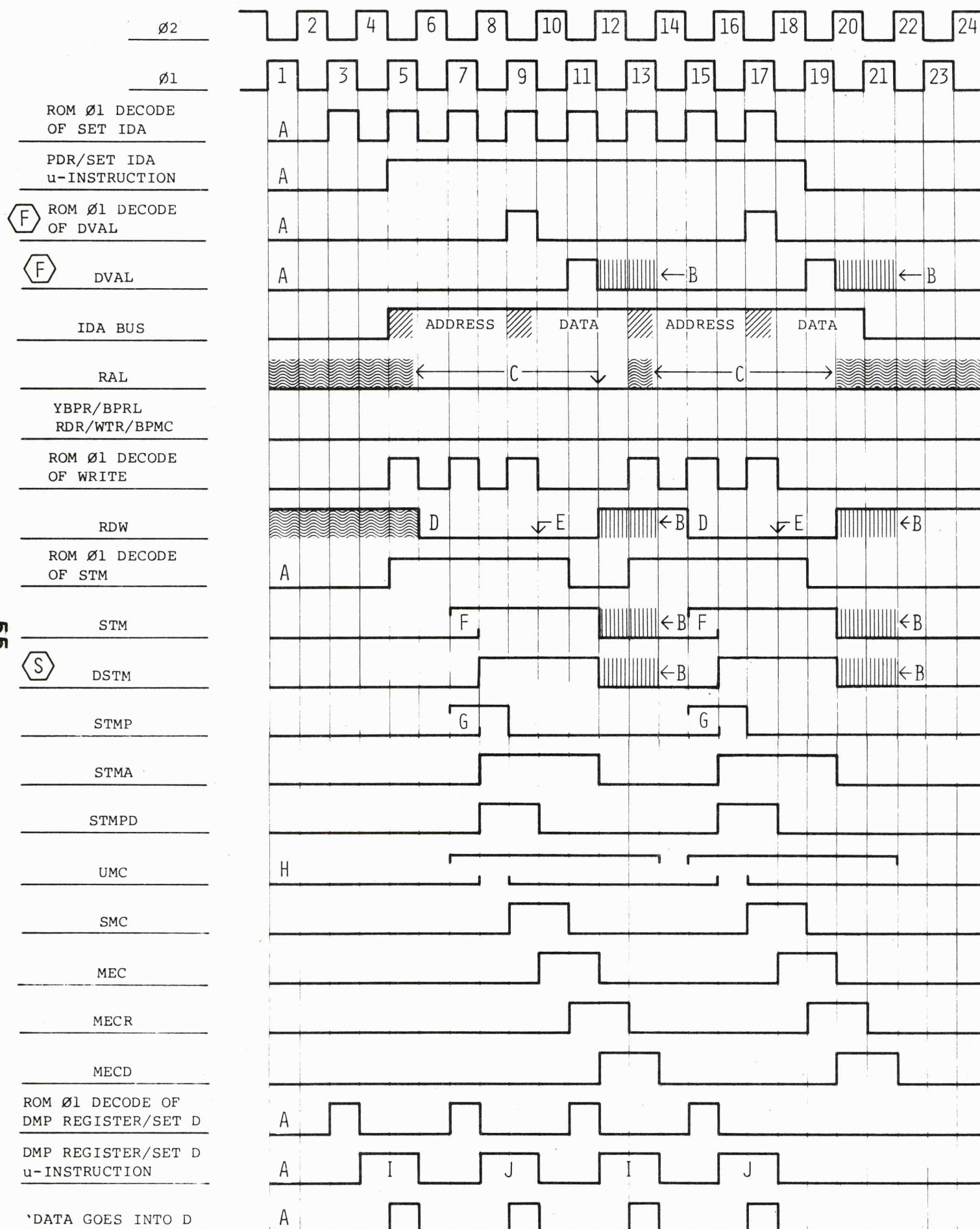


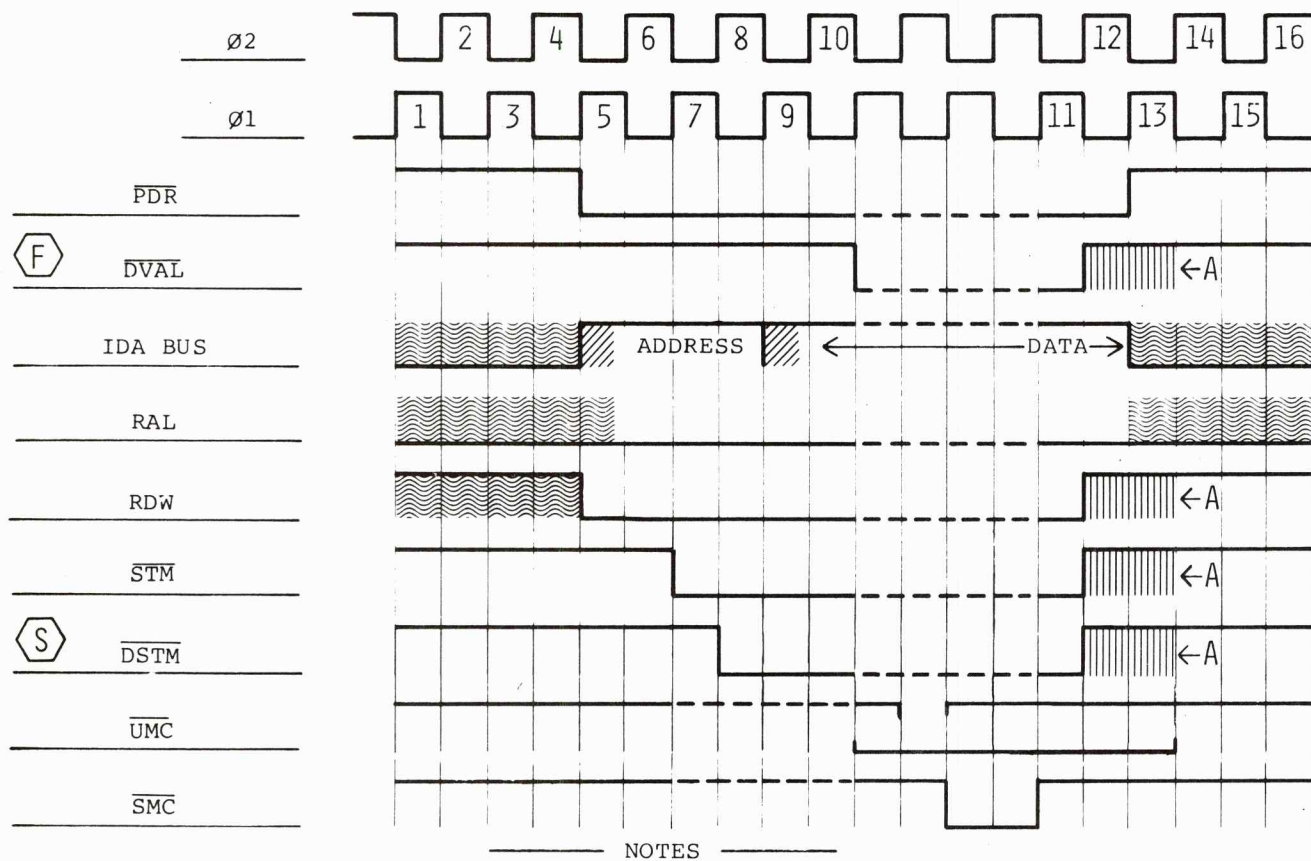
FIG 19-9

WRITE TO A BPC REGISTER

WRITE MEMORY

- NOTES
- A. THIS WAVE FORM PRESENT ONLY IF THE BPC IS THE ORIGINATOR OF THE MEMORY CYCLE.
 - B. ACTIVE PULL-UP DURING MECD ON DVAL, RDW, STM AND DSTM.
 - C. MEANING DURING THIS TIME, BUT LATCHED DURING STMA ONLY. SHOULD BE LOOKED AT DURING STM ONLY.
 - D. WAVEFORM SHOWN ASSUMES BPC AS THE ORIGINATOR OF THE MEMORY CYCLE. IF THE ORIGINATOR WERE AN EXTERNAL AGENCY, RDW COULD TRANSITION TOGETHER WITH STM.
 - E. EARLIEST RELEASE OF RDW IF AN EXTERNAL AGENCY WERE THE ORIGINATOR OF THE MEMORY CYCLE.
 - F. TRANSITIONS IMMEDIATELY AT THE START OF 01 IF THE BPC IS THE ORIGINATOR OF THE MEMORY CYCLE.
 - G. FOLLOWS STM.
 - H. UMC NEED NOT BE USED IF SMC IS GIVEN BY THE ORIGINATOR EXACTLY AS SHOWN, INSTEAD. HOWEVER, IF UMC IS GIVEN AS SHOWN, IT WILL RESULT IN SMC AS SHOWN.
 - I. UMC IS IDLE IF THE BPC IS THE ORIGINATOR.
 - J. FOR ADDRESS.

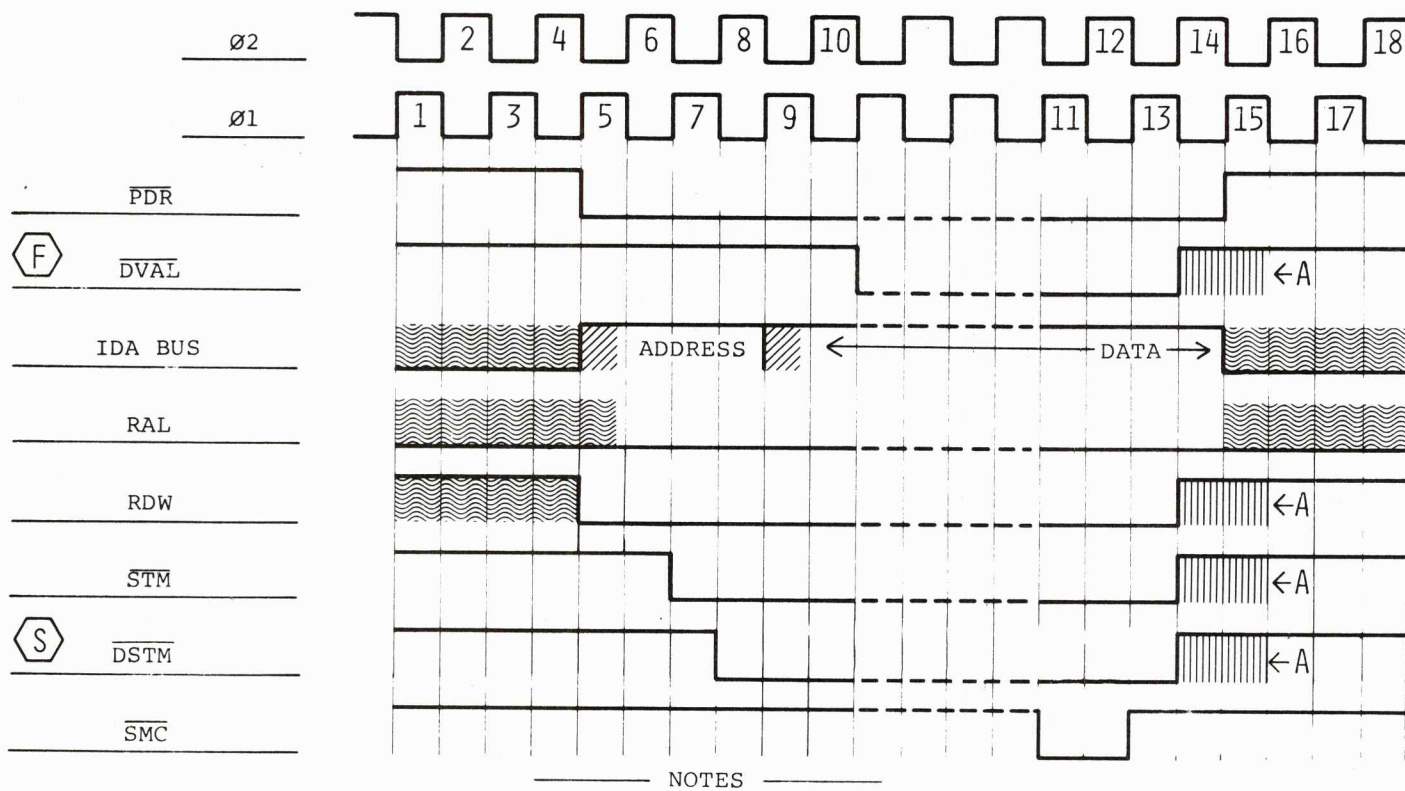
GENERALIZED 4-STATE BPC WRITE MEMORY CYCLE WITHOUT HANDSHAKE



- NOTES
- ACTIVE PULL-UP DURING MECD.
 - DOTTED LINES REPRESENT ZERO OR ANY INTEGRAL NUMBER OF ø1-ø2 PAIRS.
 - NOTE THAT \overline{UMC} COULD EQUAL \overline{STM} .

FIG 19-10

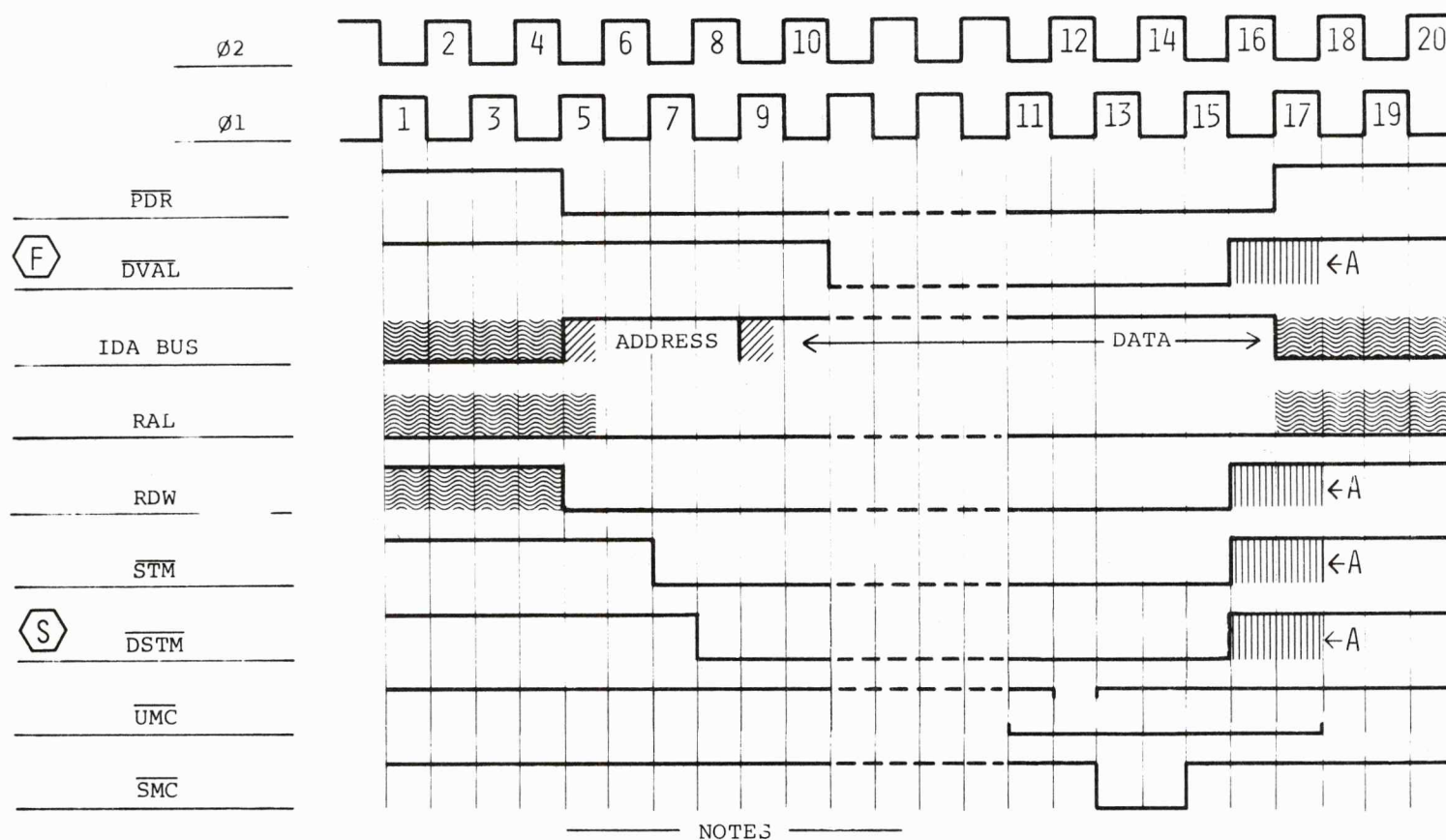
56

GENERALIZED 5-STATE BPC WRITE MEMORY CYCLE WITH HANDSHAKE
(LEADING EDGE OF DVAL USED TO INITIATE SMC)

- NOTES
- ACTIVE PULL-UP DURING MECD.
 - DOTTED LINES REPRESENT ZERO OR ANY INTEGRAL NUMBER OF ø1-ø2 PAIRS.
 - NOTE THAT \overline{SMC} CANNOT EQUAL \overline{DVAL} ; \overline{DVAL} LASTS TOO LONG.

FIG 19-11

GENERALIZED 6-STATE BPC WRITE MEMORY CYCLE WITH HANDSHAKE (LEADING EDGE OF DVAL USED TO INITIATE UMC)



- NOTES
- A. ACTIVE PULL-UP DURING MECD.
 - B. DOTTED LINES REPRESENT ZERO OR ANY INTEGRAL NUMBER OF Ø1-Ø2 PAIRS.
 - C. NOTE THAT UMC COULD EQUAL DVAL.

FIG 19-12

BUS REQUEST AND BUS GRANT DURING MEMORY CYCLES

EQUIVALENT STRUCTURE OF TWO
BACK TO BACK MEMORY CYCLES

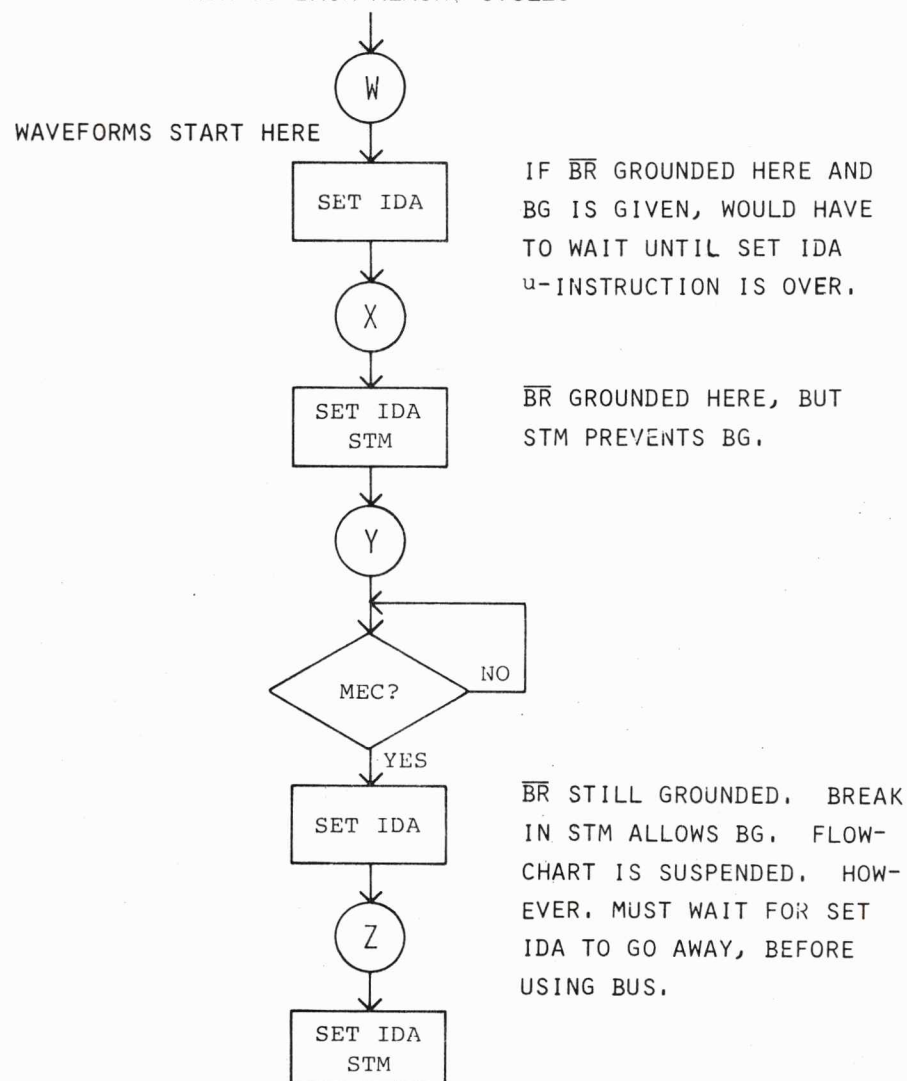


FIG 19-13-1

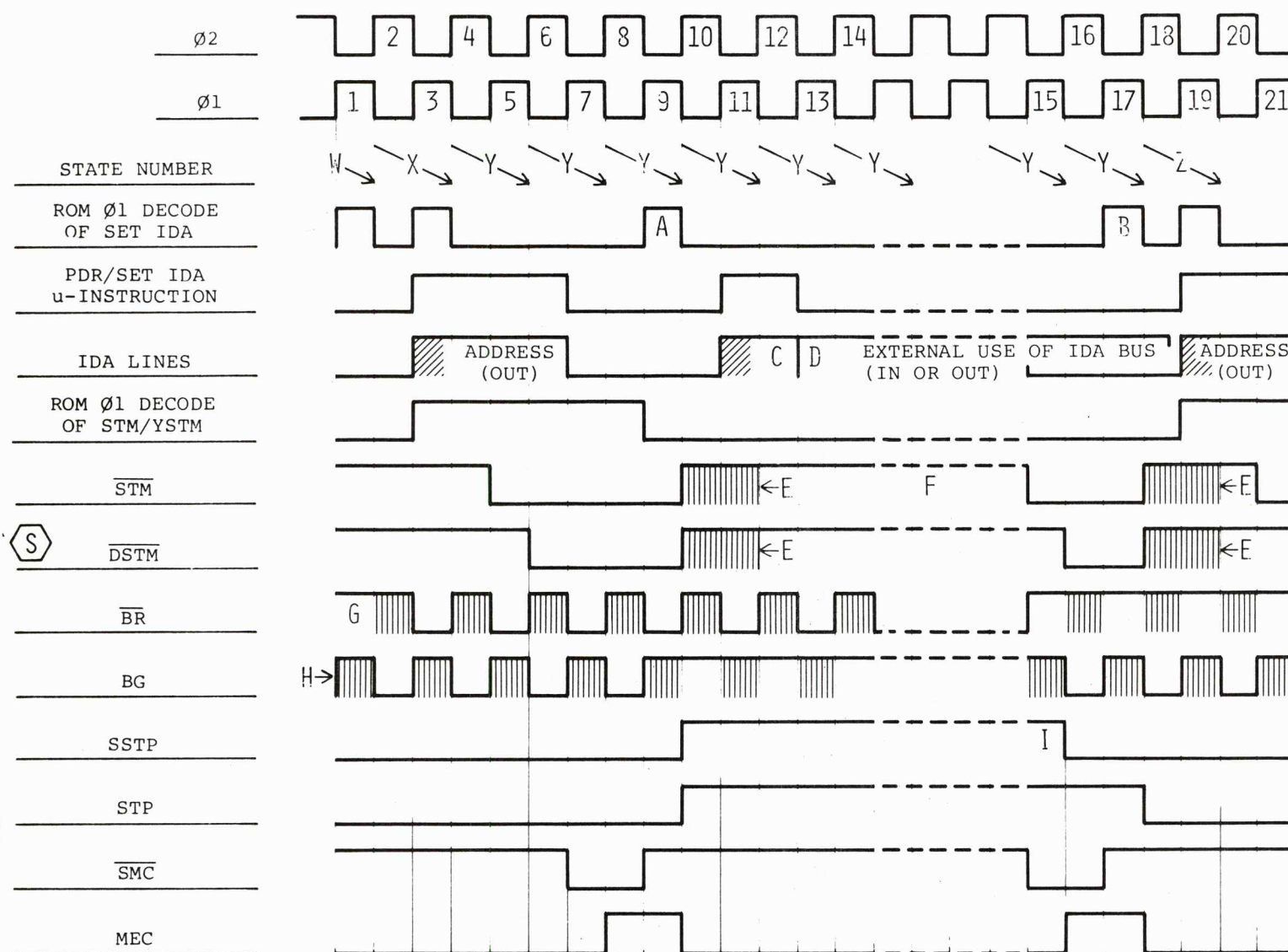
4-STATE
WRITE
MEMORY CYCLE

5-STATE
WRITE
MEMORY CYCLE

6-STATE
WRITE
MEMORY CYCLE

\overline{BR} AND BG
DURING
MEMORY CYCLES

BUS REQUEST AND BUS GRANT DURING MEMORY CYCLES



NOTES

- THIS IS THE FIRST OF TWO SET IDA'S. THE SECOND ONE IS NOT DECODED DUE TO THE GRANTING OF THE BUS. NOTE THAT EXTERNAL USE OF THE BUS MUST WAIT UNTIL THE EFFECT OF THIS SET IDA IS OVER.
- THIS IS A RE-DECODING OF THE SAME SET IDA MENTIONED IN NOTE A. IN THIS WAY THE SET IDA/SET IDA-STM SEQUENCE IS PRESERVED.
- MEMORY DATA. IT SO HAPPENS WE SHOW DATA FOR A READ CYCLE, AS OPPOSED TO THAT FOR A WRITE CYCLE. IT DOESN'T MATTER.
- EARLIEST POSSIBLE USE OF THE BUS.
- ACTIVE PULL-UP ON \overline{STM} AND \overline{DSTM} DURING MECD.
- DOTTED LINES REPRESENT AN INDEFINITE INTERRUPTION.
- IF AN IOC IS IN THE SYSTEM, IT WILL PRE-CHARGE \overline{BR} DURING ø2.
- ACTIVE PULL-UP OF BG DURING ø1.
- SSTP ENDS SOONER THAN STP. THIS ALLOWS A POSSIBLE SET IDA TO BE RE-DECODED 1 STATE PRIOR TO ALL OTHER RESUMED ROM ACTIVITY.

FIG 19-13-2

TABLE OF BPC ROM CONTENTS

MICRO-INSTRUCTION	ROM LINE	GIVEN IN ANY OF THESE GROUPS	WHEN IN ANY OF THESE STATES	PROVIDED ALL OF THESE CONDITIONS ARE MET
INTE	1	ABCDEFGHJKLM	0	MEC
SET IDA	2	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	RDR
	3	ABCDEFGHJKLM	1 6 7 8 10 14	SSTP
	4	ABCDEFG	1 3 8 9 10	MEC · SSTP
	5	ABCDEFG	2 4 14	SSTP
	6	ABCDEFGHJKLM	9 10	SSTP
	7	EFH	2 9	SSTP
	8	HM	1 3 8 9 10	SSTP
SET D	9	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	RDR · DRQ
	10	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 7 · WTR
	11	ABCDEFGHJKLM	6 14	STP
	12	ABCDEFGHJKLM	1 8	STP
	13	ABCDEFG	1 3 8	BPRL · MEC · STP
	14	ABD	1 3 8	IND · MEC · STP
	15	ABCGIJK	7 8	STP
	16	CGI	4	
	17	ABCDEFGHJKLM	9	MEC
	18	H	2 9	STP
	19	H	2 9	STP
DMP IDA	20	ABCDEFG	3	BPRL · MEC
	21	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	WTR
	22	D	2	STP
	23	DEF	2	IND · STP
	24	H	4	BPRL · MEC
	25	ABCDEFGHJKLM	0	BPRL · MEC
DMP T	27	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 6 · RDR
	30	D	3 4 5	IND · MEC
	31	EFH	9	MEC
STM	28	ABD	2	STP
	29	AK	4 5	STP
	32	ABCDEFGHJKLM	10	STP
	33	AC	2	STP
	36	ABCDEFG	2	IND · STP

FIG 20-1

TABLE OF BPC ROM CONTENTS (CONT.)

MICRO-INSTRUCTION	ROM LINE	GIVEN IN ANY OF THESE GROUPS	WHEN IN ANY OF THESE STATES	PROVIDED ALL OF THESE CONDITIONS ARE MET
STM (CONT.)	37	DEF	7 8	STP
	40	HM	3 9	STP
SET T	34	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 6 · WTR
	35	ABCDEFGHJKLM	0	MEC
	38	DM	2	STP
	39	F	2	IND · STP
	41	F	2	IND · STP
ADM	42	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 2 · RDR
	43	ABCDEFGHJKLM	0	
	45	ABCDEFGHJKLM	1	
DVAL	44	ABCDEFGHJKLM	9	MEC
SET P	46	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 16 · WTR
	47	AB	3	IND · MEC
	48	ABD	7	STP
	49	ABCDEFGHJKLM	9	MEC
	50	E	2 4 14	IND · STP
	51	HIJKM	6 14	STP
	52	H	4	MEC
	53	ABCDEFGHJKLM	14	
DMP P	54	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 16 · RDR
	55	CEFGHI	8	
	56	ABD	9	SKP · MEC
	57	ABCG	7 8	SKP · STP
DMP R	58	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 17 · RDR
	59	ABCDEFGHJKLM	0	MEC
	61	H	2	STP
WRITE	60	C	2 4	IND · STP
	64	ABCDEFGHJKLM	9	MEC
	65	DEF	7 8	STP
SET R	62	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 17 · WTR
	63	ABCDEFG	6 14	STP
	66	H	9 10	STP

FIG 20-2

BR AND BG DURING MEMORY CYCLES (CONT)

BPC ROM CONTENTS (CONT)

TABLE OF BPC ROM CONTENTS (CONT.)

MICRO-INSTRUCTION	ROM LINE	GIVEN IN ANY OF THESE GROUPS	WHEN IN ANY OF THESE STATES	PROVIDED ALL OF THESE CONDITIONS ARE MET
SET S	67	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 12·WTR
	68	ABCDEFGHJKLM	0	MEC
	69	ABD	3	$\overline{\text{IND}} \cdot \text{MEC} \cdot \text{STP}$
	70	IJK	2	$\overline{\text{STP}}$
SSE	71	J	3 9	$\overline{\text{STP}}$
SYNC	72	AB	3 9	$\overline{\text{IND}} \cdot \text{MEC}$
	73	ABCDEFGHJKLM	9	MEC
	76	EG	2 4	$\overline{\text{IND}}$
	77	H	4	MEC
	80	IJK	2	
	81	JKM	2	
DMP EX/OV	75	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	(M-ADD 4 + M-ADD 5)·RDR
SET OV	78	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 5·WTR
	79	I	9 10	$\text{OQ} \cdot \overline{\text{STP}}$
SET EX	82	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 4·WTR
	83	I	9 10	$\text{EQ} \cdot \overline{\text{STP}}$
DMP PAD	84	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	(M-ADD 0 + M-ADD 1 + M-ADD 2 + M-ADD 3)·RDR
	85	AB	3 9	$\overline{\text{IND}} \cdot \text{MEC}$
	86	ACK	9	MEC
	87	DM	7	$\overline{\text{STP}}$
	88	HIJKM	1 6	$\overline{\text{STP}}$
	89	ABCDEFGHJKLM	1	$\overline{\text{STP}}$
	90	BG	7 8	$\text{SKP} \cdot \overline{\text{STP}}$
	91	ABD	9	$\text{SKP} \cdot \text{MEC}$
RTP	92	ABCDEFGHJKLM	0	
NS0	93	ABCDEFGHJKLM	0	$\overline{\text{MEC}}$
	96	AC	5 6	
	97	ABCDEFGHJKLM	10	
ADS	94	HI	4 5 6 14	SKP
	95	H	4 5 6 14	

FIG 20-3

TABLE OF BPC ROM CONTENTS (CONT.)

MICRO-INSTRUCTION	ROM LINE	GIVEN IN ANY OF THESE GROUPS	WHEN IN ANY OF THESE STATES	PROVIDED ALL OF THESE CONDITIONS ARE MET
UPDOE	98	K	10	
	99	A	5 6	
NS9	100	C	4	$\overline{\text{MEC}}$
	101	ABCDEFGHJKLM	9	
DMP ALU	102	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 11·RDR
	103	ABCDEFG	5 6	$\overline{\text{STP}}$
	106	ABDJKM	8	$\overline{\text{STP}}$
	107	HIJKM	10	
NS7	104	BG	5 6	$\overline{\text{IND}} \cdot \text{MEC}$
	105	D	3 4 5	
NS4	108	CF	2	$\overline{\text{IND}}$
	109	HM	4	$\overline{\text{MEC}}$
SET B	111	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 15·WTR
	114	ACK	5	BQ
	115	IJK	9 10	$\text{BQ} \cdot \overline{\text{STP}}$
NS6	112	JK	3	CTQ
	113	M	2	
	116	K	3	SYNCQ
	117	M	2	
DMP B	118	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 15·RDR
	119	J	2 4	$\text{BQ} \cdot \overline{\text{STP}}$
	122	C	4	$\text{BQ} \cdot \overline{\text{STP}}$
NS10	120	ABCG	7 8	$\overline{\text{GNI}}$
	121	HIJM	6 14	
	124	IJK	6 14	
	125	ABCDEFGHJKLM	14	$\overline{\text{IND}}$
	128	EG	2 4 9	
DMP A	123	ABCDEFGHJKLM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 14·RDR
	126	J	2 4	$\text{AQ} \cdot \overline{\text{STP}}$
	127	C	4	$\text{AQ} \cdot \overline{\text{STP}}$

FIG 20-4

TABLE OF BPC ROM CONTENTS (CONT.)

MICRO-INSTRUCTION	ROM LINE	GIVEN IN ANY OF THESE GROUPS	WHEN IN ANY OF THESE STATES	PROVIDED ALL OF THESE CONDITIONS ARE MET
NS2	129	ABD	3	IND·MEC
	132	CEFG	3	MEC
	133	I	2	$\overline{\text{SYNCQ}}$
	136	M	2	$\overline{\text{SYNCQ}} \cdot \overline{\text{GNI}}$
	137	DM	4	MEC
SET A	130	ABCDEFGHGIJKM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 14·WTR
	131	ACK	5	AQ
	134	IJK	9 10	AQ· $\overline{\text{STP}}$
DMP ST	135	ABCDEFGHGIJKM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 13·RDR
	138	ABCDEFGHGIJKM	14	$\overline{\text{STP}}$
SET I	139	ABCDEFGHGIJKM	0 1 2 3 4 5 6 7 8 9 10 14	M-ADD 13·WTR
	142	ABCDEFGHGIJKM	0	$\overline{\text{MEC}}$
NS3	140	ABCDEFG	3	$\overline{\text{MEC}}$
	141	DM	5 6	GNI
	144	J	3 4 5	$\overline{\text{CTQ}}$
SYNC F	143	ABCDEFGHGIJKM	0	

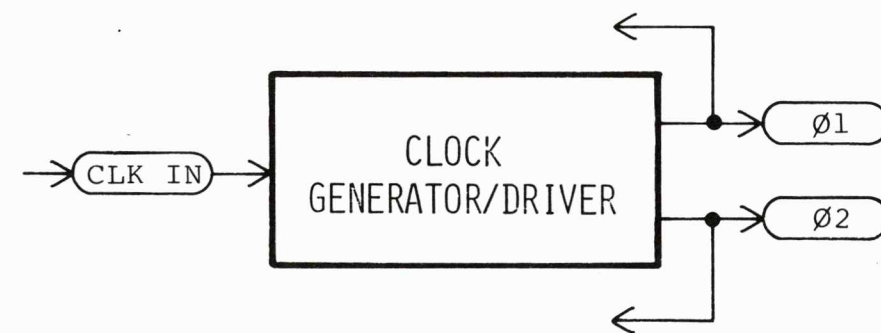
FIG 20-5

BPC ROM
CONTENTS (CONT)

BPC ROM
CONTENTS (CONT)

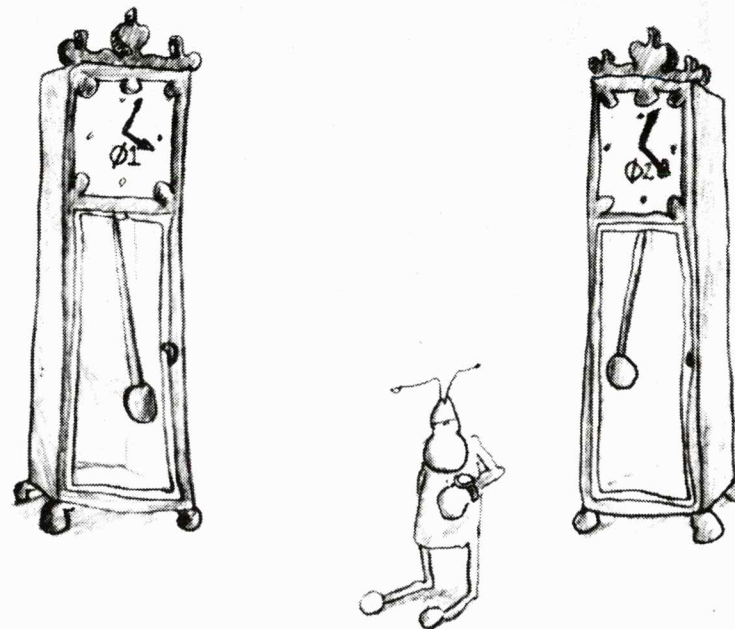
BPC ROM
CONTENTS (CONT)

OVERVIEW OF THE CLOCK GENERATOR/DRIVER CIRCUIT (16-BIT VERSION ONLY)



THIS CIRCUIT OCCURS
IN THE 16-BIT VERSION ONLY

FIG 21-1-S



THE SYSTEM CLOCKS

SECTION 21

Figure 21-1 is an overview of the Clock Generator/Driver that occurs in the 16-bit version only. A twice-speed TTL level clock is used to drive the Clock Generator; it in turn produces phase one and phase two.

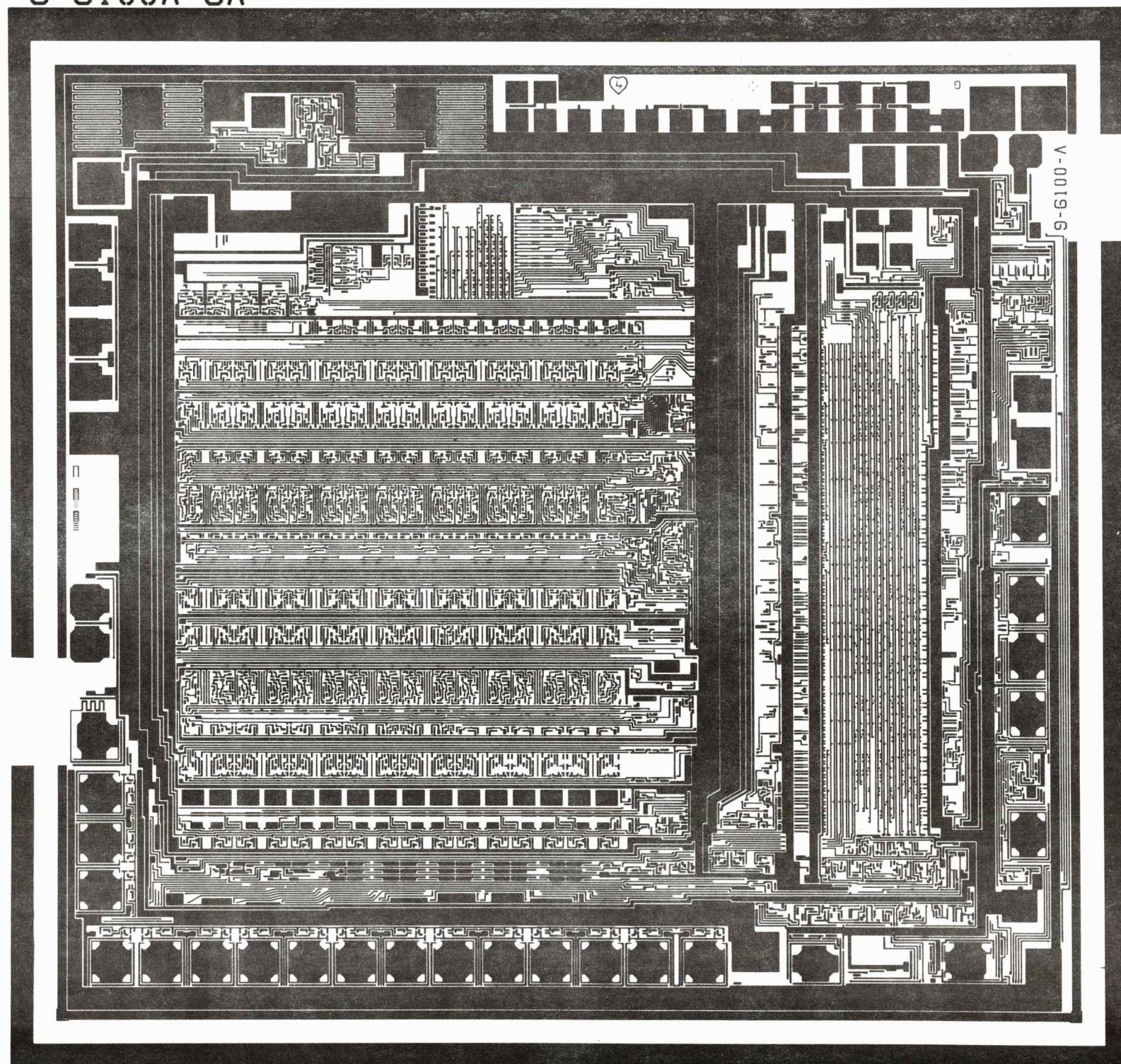
Figure 21-2 shows the details of the Clock Generator/Driver circuit. The Driver portion of the Clock Generator is adequate to supply the phase one/phase two needs of all the entities in the hybrid processor system.

**CLOCK
GENERATOR/DRIVER**

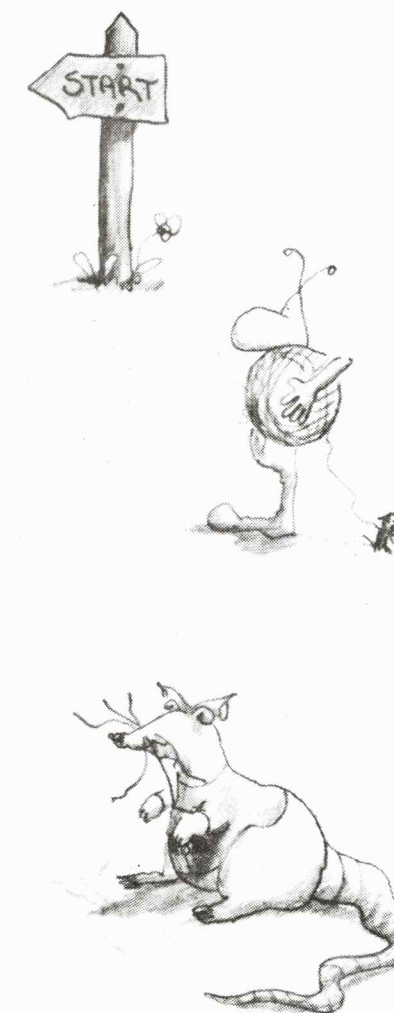


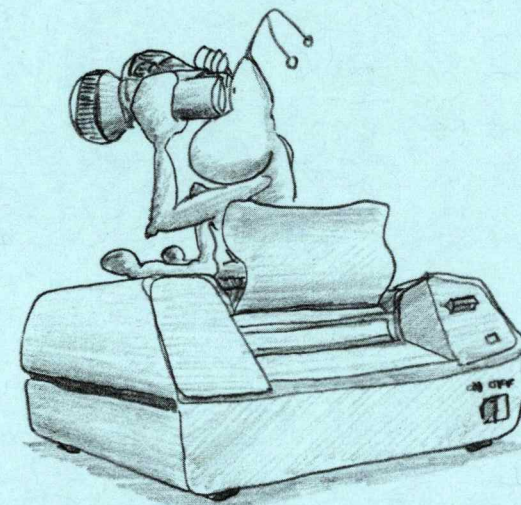
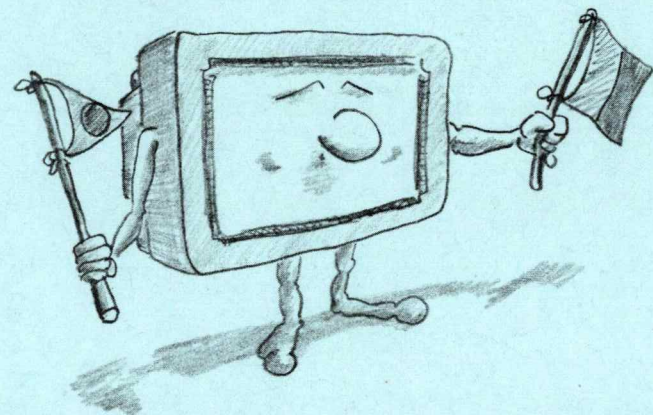
63

9-6100A-6A



2-20-76





IOE

SECTION 1

Figures 1-S and 1-F are block diagrams of the 16-bit and 15-bit versions of the IOC, respectively. A DMP IDA micro-instruction provides communication from the IDA Bus to the internal IDC Bus in the IOC. A SET IDA micro-instruction provides communication from the IOC to the IDA Bus; SET IDA drives the IDA Bus according to the contents of the O Register, which in turn is set with a SET O micro-instruction.

As in the BPC, an Address Decode section and associated latches detect the appearance of an IOC-related register address. Such an event results in the address being latched and sent to the Bus Control ROM as qualifier information.

There are two main ROM's in the IOC. These are the Bus Control ROM and the Instruction Control ROM. The Bus Control ROM is responsible for generating and responding to activity between the IOC and the IDA and IDC busses. This class of activity consists of memory cycles, I/O Bus cycles, interrupt polls, interrupt requests, and requests for DMA. The Instruction Control ROM is responsible for recognizing fetched IOC machine-instructions, and for implementing the algorithms that accomplish those instructions. Frequently, the Bus Control ROM will undertake activity on behalf of the Instruction Control ROM. These two ROM's are physically merged, and share a common set of decodable micro-instructions.

However, each of the two ROM's has its own state-counter. For each ROM, the next state is explicitly decoded by each current state. Section 22 contains flow charts depicting Instruction Controller activity.

The I register serves a function similar to that of the I register in the BPC. It serves as a repository to hold the fetched machine-instruction and to supply that instruction to Instruction Decode. Instruction Decode generates asynchronous control lines that are similar in function to those of the BPC. Instruction Decode also

generates instruction qualifiers that represent the machine-instruction to the ROM mechanism.

The W register is used primarily in conjunction with the execution of the "Place" and "Withdraw" machine-instructions. Each such instruction requires two memory cycles; one to get the data from the source, and one to transmit it to the destination. W serves as a place to hold the data inbetween those memory cycles.

The DMP W function is complex, and is implemented by a DMP W and Crossover Network. If the place or withdraw operation is for the entire word, the crossover function is not employed, and the pairs of signals OLB, DLB, and OMB, DMB, work together to implement a standard 16-bit DMP W. However, a byte-oriented place or withdraw instruction involves the dumping of only a single byte of W onto the IDC Bus. This is done in the following combinations: least-significant byte of W to either half of the IDC Bus; and, most-significant byte of W to either half of the IDC Bus. The exact mode of operation during a DMP W is determined by W Register Control on the basis of asynchronous control lines from Instruction Decode and information from the C and D registers. The information from the C and D registers is the occasion for a difference between the 15 and 16-bit versions. The 16-bit version is sensitive to C(0) and D(0), while the 15-bit version responds to C(15) and D(15).

Another use of W occurs during an interrupt. During an interrupt poll the response of the requesting peripheral(s) is loaded into the least-significant half of W. These eight bits represent the eight peripherals on the currently active (or enabled) level of interrupt. Each peripheral requesting interrupt service during the poll will have a one in its corresponding bit. This 8-bit pattern is fed to a Select Code Priority Resolver and 3 LSB Interrupt Vector Generator. That circuitry identifies the highest numbered select code requesting

service (should there be more than one) and generates the three least-significant bits of binary code that correspond to that peripheral's select code. The next most-significant bit corresponds to the level at which the interrupt is being granted, and it is available from the interrupt circuitry in the form of the signal PHIR.

The interrupt vector is based on the eight least-significant bits from W (as encoded into three bits by the priority resolver), the bit corresponding to PHIR, and the 12 bits contained in the Interrupt Vector Register (IV). Thus, when an interrupt is granted the complete interrupt vector is placed on the IDC Bus by simultaneously giving the following micro-instructions: EPR, DMP ISC, UIG, and DMP IV.

The C and D registers are the pointer registers used for "Place" and "Withdraw" operations. In the 16-bit version these registers operate in conjunction with the 1-bit registers CB and DB (respectively) to provide word-oriented or byte-oriented addresses for the firmware-managed stacks. Each of the C and D registers is equipped with a 16-bit increment and decrement network for changing the value of the pointer. A means is also included for updating CB and DB, as if they were 17th bits. Whether to increment or decrement is controlled by the asynchronous control lines.

In the 15-bit version C and D serve the same function, although in a slightly different manner, and without the benefit of CB and DB. In this scheme bit 15 (the MSB) of each pointer register toggles most rapidly for byte-oriented increments or decrements. The lower 15 bits serve as a word address. The INC/DEC Network serves only the lower 15 bits. Dedicated circuitry in C and D Register Control determine when to toggle the 16th bit of a pointer register.

The DMA Memory Address (DMAMA) and DMA Count (DMAC) registers are similar to the C and D registers, except that DMAMA always increments, and that DMAC al-

ways decrements. These two registers are used in conjunction to identify the destination or source address in memory of each DMA transfer, and to keep a count of the number of such transfers so far.

In both versions these registers each have 16 bits. There are some differences, however. The increment and decrement networks are all 16-bit circuits, except for DMAMA in the 15-bit version. That increment network is only a 15-bit circuit, since there would never be a need to increment the most significant bit of a DMA address. Instead, that bit is used to indicate the direction of DMA transfers. Since the 16-bit version requires that bit for addressing, DMA direction must be indicated by a separate entity. This is done with the 1-bit DMAD register. It is set and cleared as the result of machine-instructions dedicated to that purpose. In either version the signal DR/WQ (DMA Read/Write Qualifier) serves as an internal indicator to the IOC concerning the direction of DMA transfers. In the 15-bit case DR/WQ is derived from bit 15 of DMAMA, and in the 16-bit case, from DMAD.

Two separate mechanisms are provided for the storage of peripheral select codes. The DMAPA register is a 4-bit register used to contain the select code of any peripheral that is engaged in DMA.

The other mechanism is a three-level stack, also four bits wide, whose uppermost level is the Peripheral Address register (PA). It is in this stack that peripheral select codes for both standard I/O and interrupt I/O are kept. The stack is managed by the interrupt circuitry.

The Peripheral Address Lines (PA Lines) reflect either the contents of DMAPA or PA, depending upon whether or not the associated I/O Bus cycles are for DMA or not, respectively. This selection is controlled by the DMA circuitry, and is implemented by the Peripheral Address Bus Controller.

Three latches control whether

SECTION 1 (CONTINUED)

or not the interrupt system is active or disabled, whether or not the DMA Mode is active or disabled, and, whether or not the Pulse Count Mode is active or disabled. Those latches are respectively controlled by these machine-instructions; EIR and DIR for the interrupt system, and, DMA, PCM, and DDR for DMA-type operations.

The interrupt circuitry is controlled by a 2-bit state-counter and ROM. The state-count is used to represent the level of interrupt currently is use. Requests for interrupt are converted into qualifiers for the ROM of the Interrupt Controller. If the interrupt request can be granted, the approved request is then represented by a change in state of that ROM, as well as by instructions decoded from that ROM, and sent to the Interrupt Grant Network. This circuitry generates the INT signal used to cause an interrupt of the BPC, and, generates an INTQ qualifier that represents the occurrence of an interrupt to the main ROM mechanism in the IOC so that an interrupt poll can be initiated.

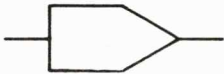


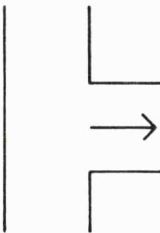
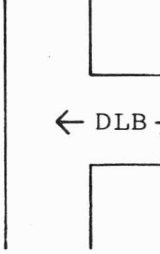
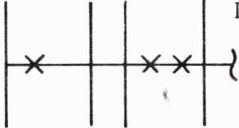




The DMA circuitry is similar in its method of control. It has a ROM controlled by a 3-bit state-counter.

Section 24 contains a variety of waveforms illustrating IOC activity in its various modes of operation. Figure 24-1 explains the convention used in the waveforms.

Numbers in circles (XXX) →

refer to the page number at which the referenced item is discussed.

NOTES FOR FIG. 1

1.  DENOTES A MICRO-INSTRUCTION DECODED IN THE ROM.
2.  AND  DENOTE ONE- AND TWO-WAY INTERCONNECTIONS TO A BUS; ALWAYS CONTROLLED BY A ROM MICRO-INSTRUCTION.
3.  DENOTES A DIRECT CONNECTION BETWEEN TWO ITEMS.
4.  DENOTES A CONNECTION BETWEEN TWO ITEMS THAT IS ACTIVE ONLY WHEN THE STATED SIGNAL IS GIVEN. SOME SUCH SIGNALS ARE ROM DECODED MICRO-INSTRUCTIONS, WHILE OTHERS ARE PRESENT THROUGHOUT AN ENTIRE EXECUTION CYCLE.
5.  DENOTES THAT THE STATED LINE REPRESENTS A DECODED CONDITION.
6.  REPRESENTS A NON-MICRO-INSTRUCTION CONTROL LINE OR SOME OTHER SIGNAL.
7.  REPRESENTS AN INPUT TERMINAL TO THE IOC.
8.  REPRESENTS AN OUTPUT TERMINAL FROM THE IOC.
9.  REPRESENTS A TERMINAL THAT IS BOTH AN INPUT AND AN OUTPUT.
10. NUMBERS IN PARENTHESES INDICATE THE NUMBER OF BITS A MECHANISM HANDLES.
11. THE LOGICAL SENSE (XXX VERSUS $\overline{\text{XXX}}$) OF THE I/O TERMINALS IS CORRECTLY INDICATED. HOWEVER, THE DRAWING IS NOT A RELIABLE INDICATOR OF THE EXACT SENSE OF THE INTERNAL SIGNALS. TYPICALLY BOTH SENSES EXIST, AND FREQUENTLY THE PHYSICAL PROXIMITY OF SIGNALS TO THEIR DESTINATIONS WAS MORE IMPORTANT IN DECIDING WHICH SENSE TO USE, RATHER THAN AGREEMENT OF LOGICAL SENSE.
BECAUSE STRICT ACCURACY IN REPORTING SIGNAL SENSES ON SUCH A GENERAL LEVEL DRAWING WOULD SHARPLY INCREASE THE NUMBER OF INTERCONNECTIONS, WITH ONLY A SLIGHT INCREASE IN USEFULNESS, WE USUALLY SHOW ONLY THE NAME OF THE SIGNAL.

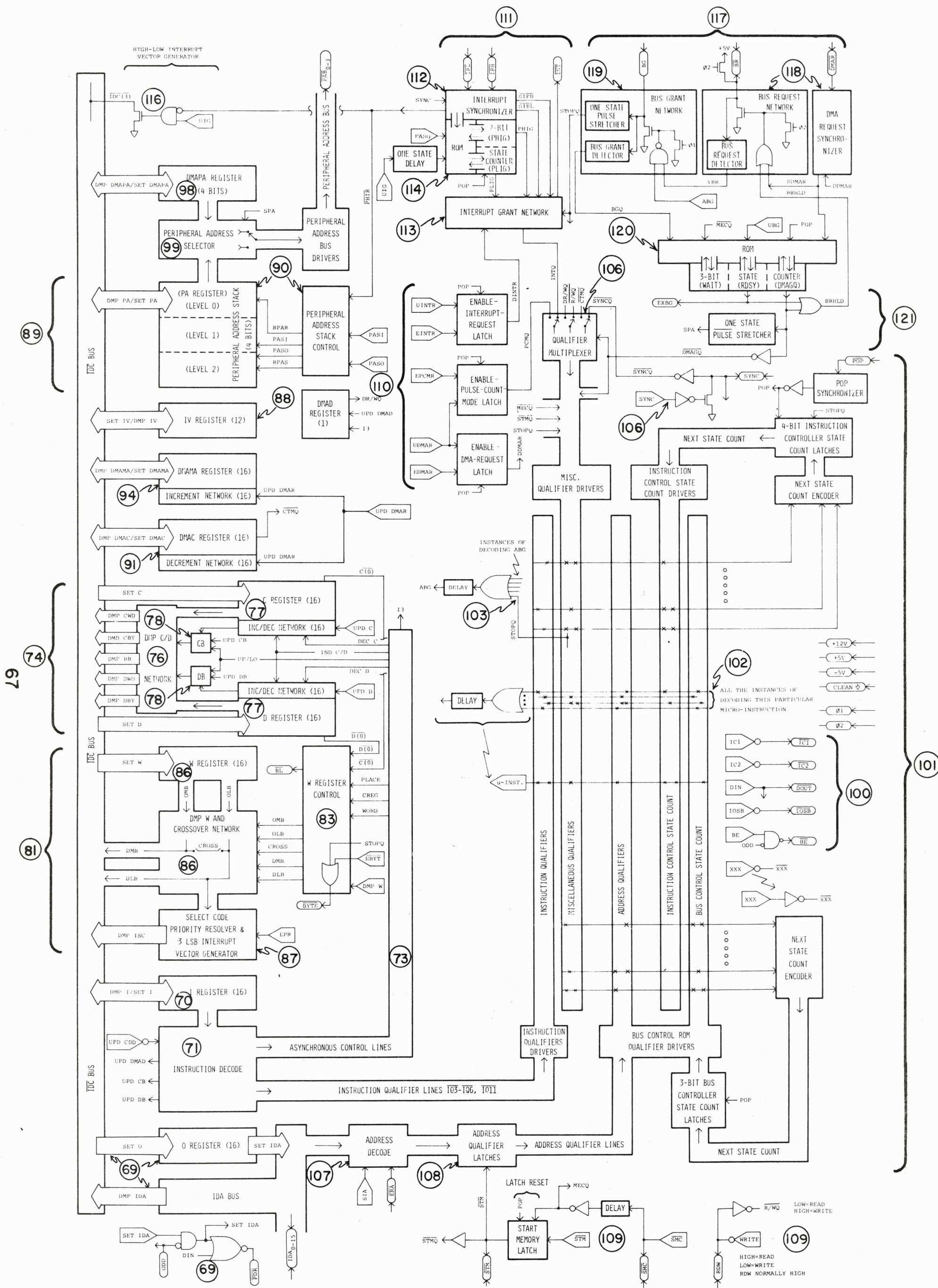


FIG I-S

IOB BLOCK DIAGRAM



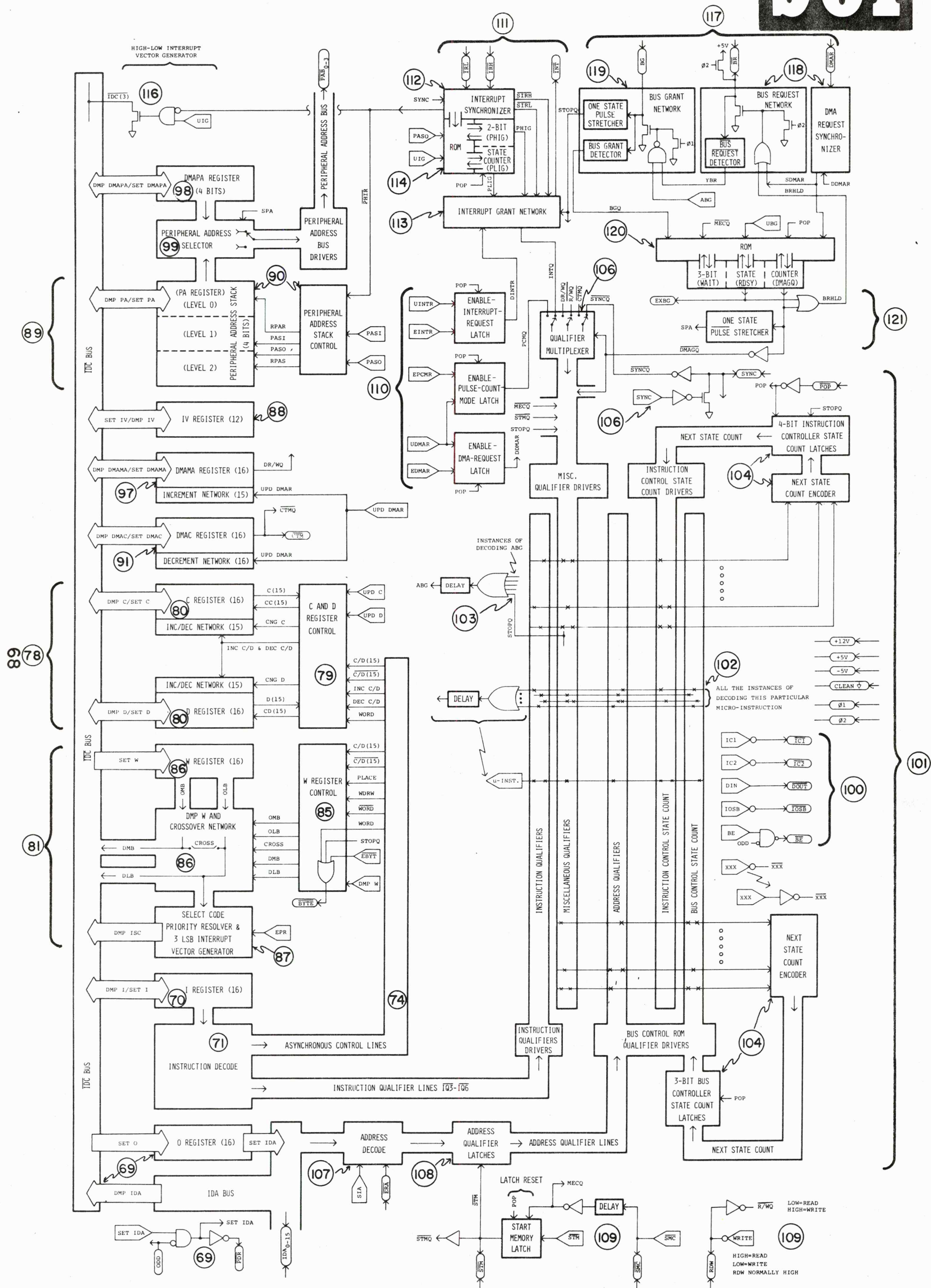


FIG 1-F

OVERVIEW OF THE IDA BUS - IDC BUS INTERCONNECTION

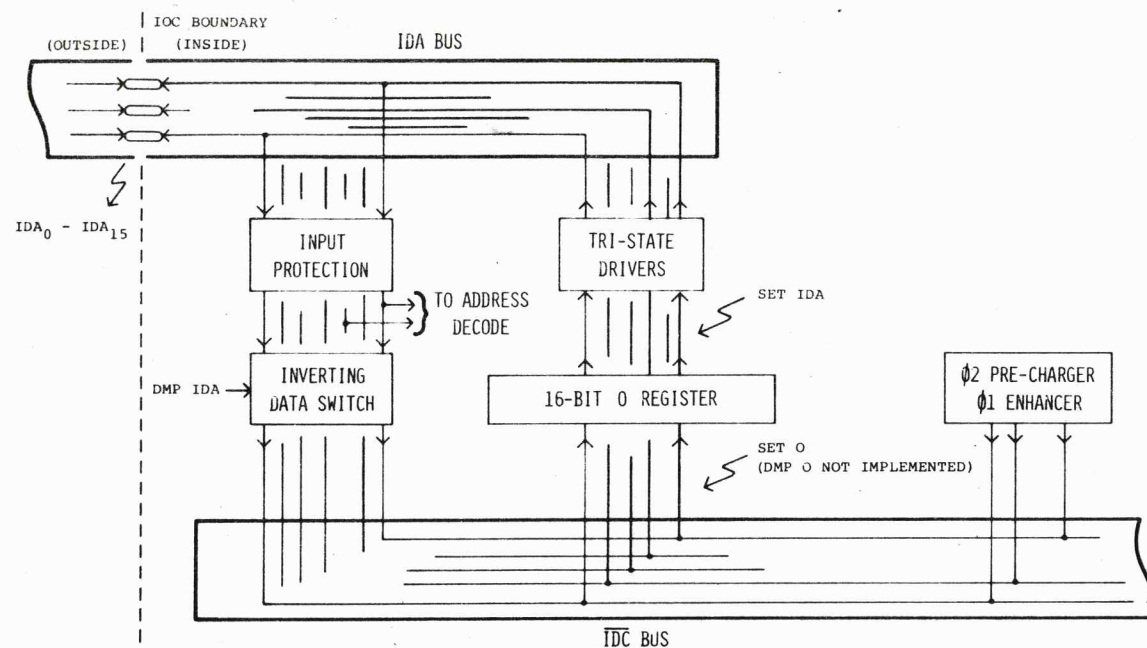
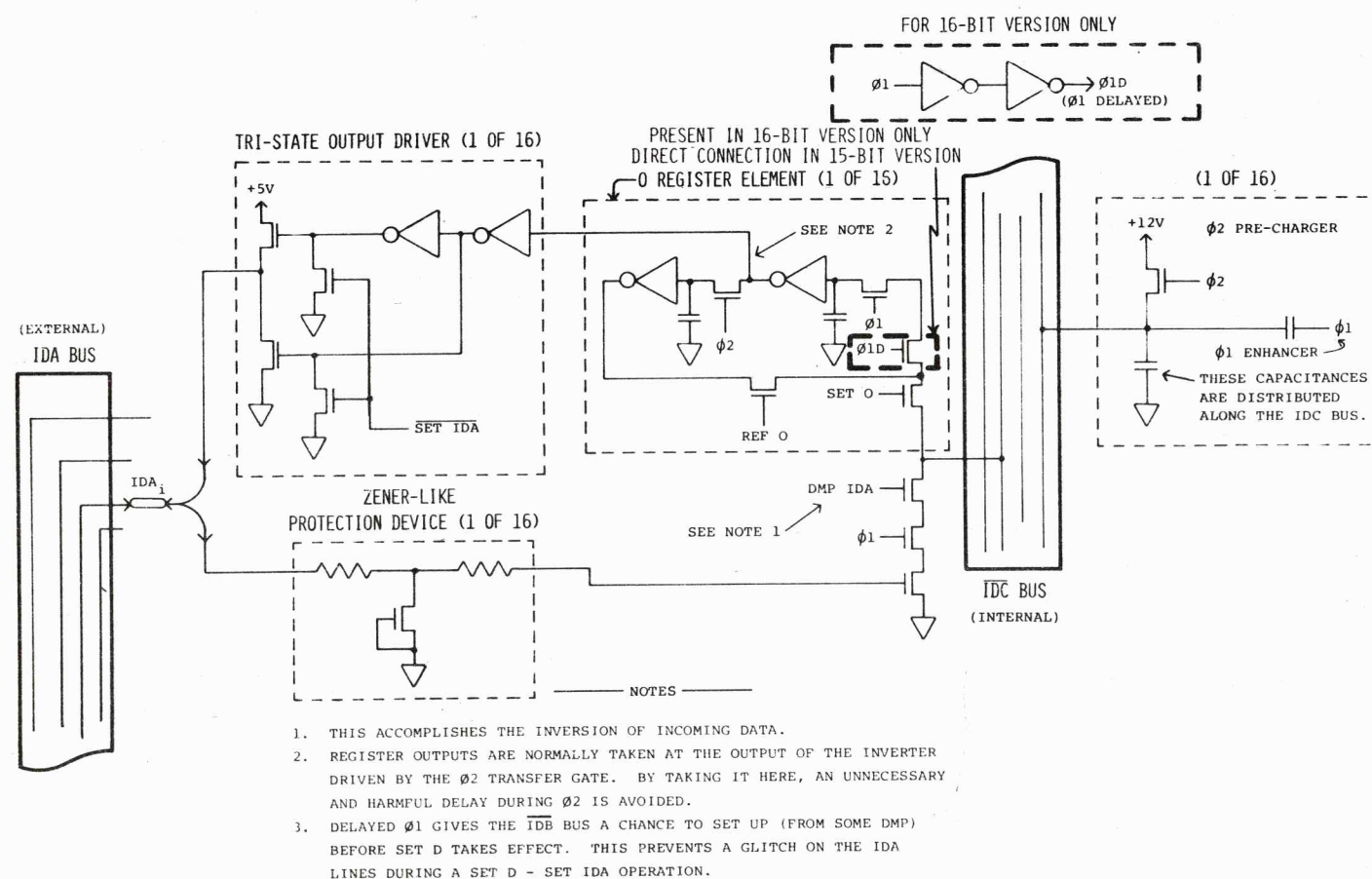


FIG 2-1

DETAILS OF THE 0 REGISTER AND THE IDA BUS DRIVERS



- NOTES
1. THIS ACCOMPLISHES THE INVERSION OF INCOMING DATA.
 2. REGISTER OUTPUTS ARE NORMALLY TAKEN AT THE OUTPUT OF THE INVERTER DRIVEN BY THE ϕ_2 TRANSFER GATE. BY TAKING IT HERE, AN UNNECESSARY AND HARMFUL DELAY DURING ϕ_2 IS AVOIDED.
 3. DELAYED ϕ_1 GIVES THE IDC BUS A CHANCE TO SET UP (FROM SOME DMP) BEFORE SET D TAKES EFFECT. THIS PREVENTS A GLITCH ON THE IDA LINES DURING A SET D - SET IDA OPERATION.

FIG 2-2

DETAILS OF THE SET IDA, PDR, DMP IDA, ODD AND SET 0 u-INSTRUCTIONS

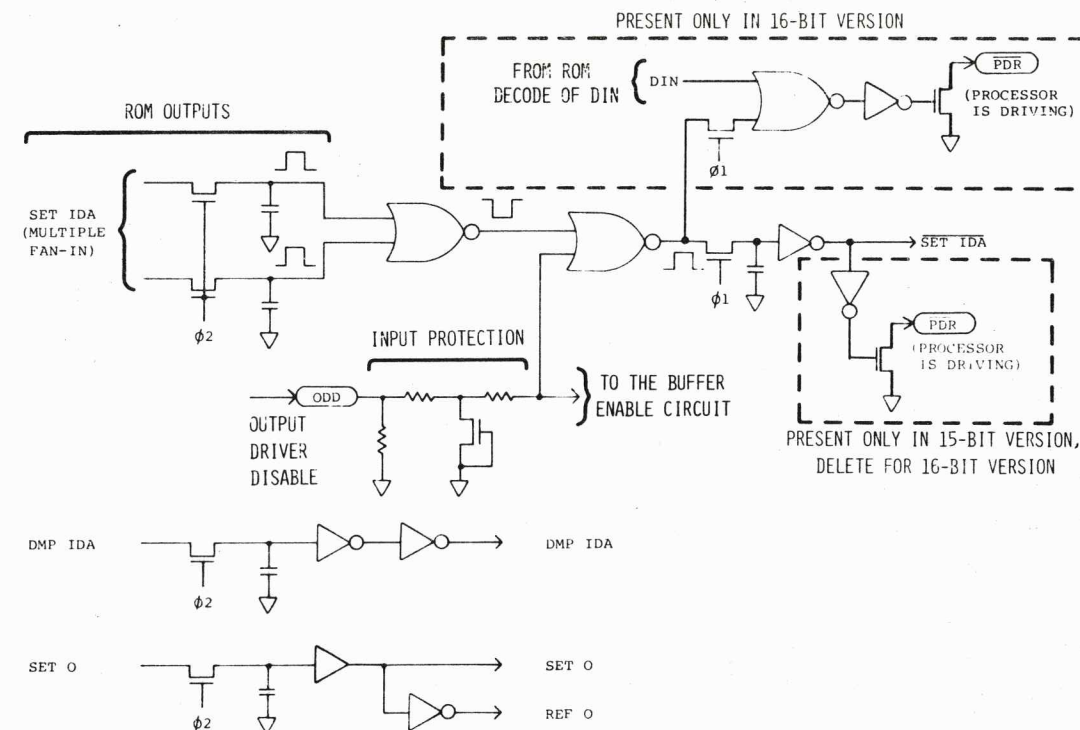


FIG 2-3

SECTION 2

Figure 2-1 is an overview of the connection between the external IDA Bus and the internal IDC Bus within the IOC. The circuitry involved is virtually identical to that of the BPC. Notice that what was called the D register in the BPC is now called the 0 register in the IOC.

Figure 2-2 illustrates the details of the 0 register, the tri-state drivers and of the circuitry that precharges the IDC Bus. The circuitry is identical to that of the BPC.

Figure 2-3 illustrates the details of PDR, the SET IDA, DMP IDA, and SET 0 micro-instructions, and includes partial details of ODD. The difference between the 15 and 16-bit versions resides in the generation of PDR. In the 15-bit version PDR is given whenever SET IDA is given; that is, whenever the IOC is driving the Bus \overline{PDR} is driven low. In the 16-bit version PDR is the OR of SET IDA and Data In (DIN). DIN is a signal associated with peripheral activity, and means that a peripheral is supplying data that has been requested of it. It is appropriate to ground \overline{PDR} when DIN is given, because the IOC arranges that the data provided by the peripheral be placed onto the IDA Bus.

IDA - IDC OVERVIEW
O REGISTER AND
IDA BUS DRIVERS

IOC BLOCK DIAGRAM

OVERVIEW OF THE I REGISTER AND ITS RELATIONSHIP WITH INSTRUCTION DECODE

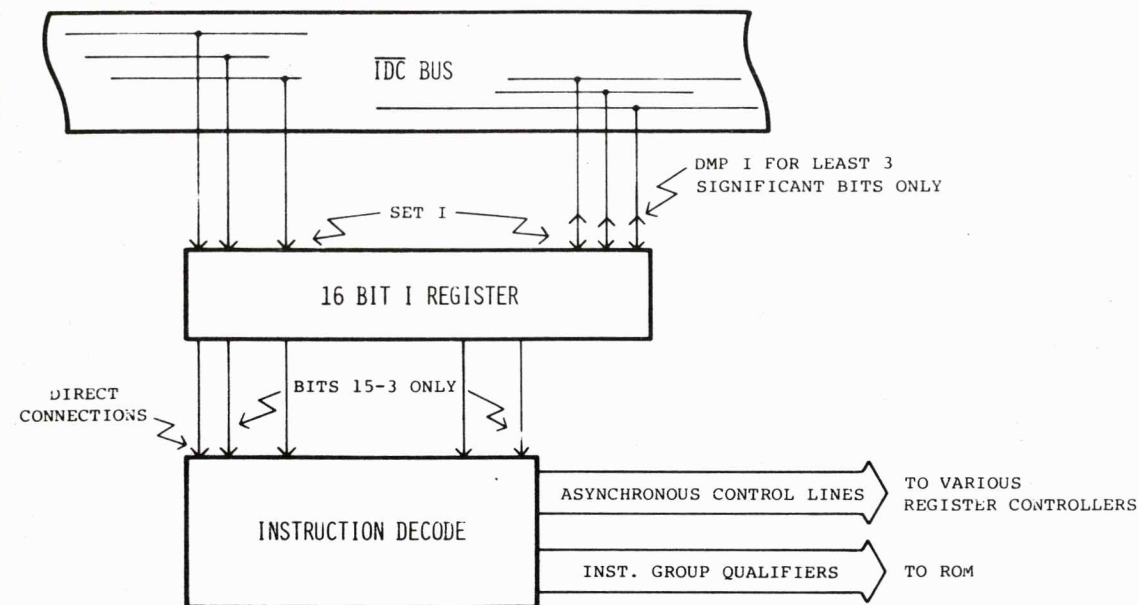


FIG 3-1

SECTION 3

Figure 3-1 is an overview of the I register and of its inter-connection with Instruction Decode. The I register serves the same basic function as in the BPC. Observe that bits 3 through 15 go via direct connection to Instruction Decode. Also observe that a DMP I micro-instruction is implemented for the three least significant bits of the I register. The DMP I is used during the execution of Place and Withdraw machine-instructions. Such instructions have their associated register encoded in the three least significant bits of the machine-instruction's bit pattern. During the execution of such a machine-instruction the three least significant bits of the I register can be dumped to form the address for a memory cycle to the associated register.

DETAILS OF THE I REGISTER

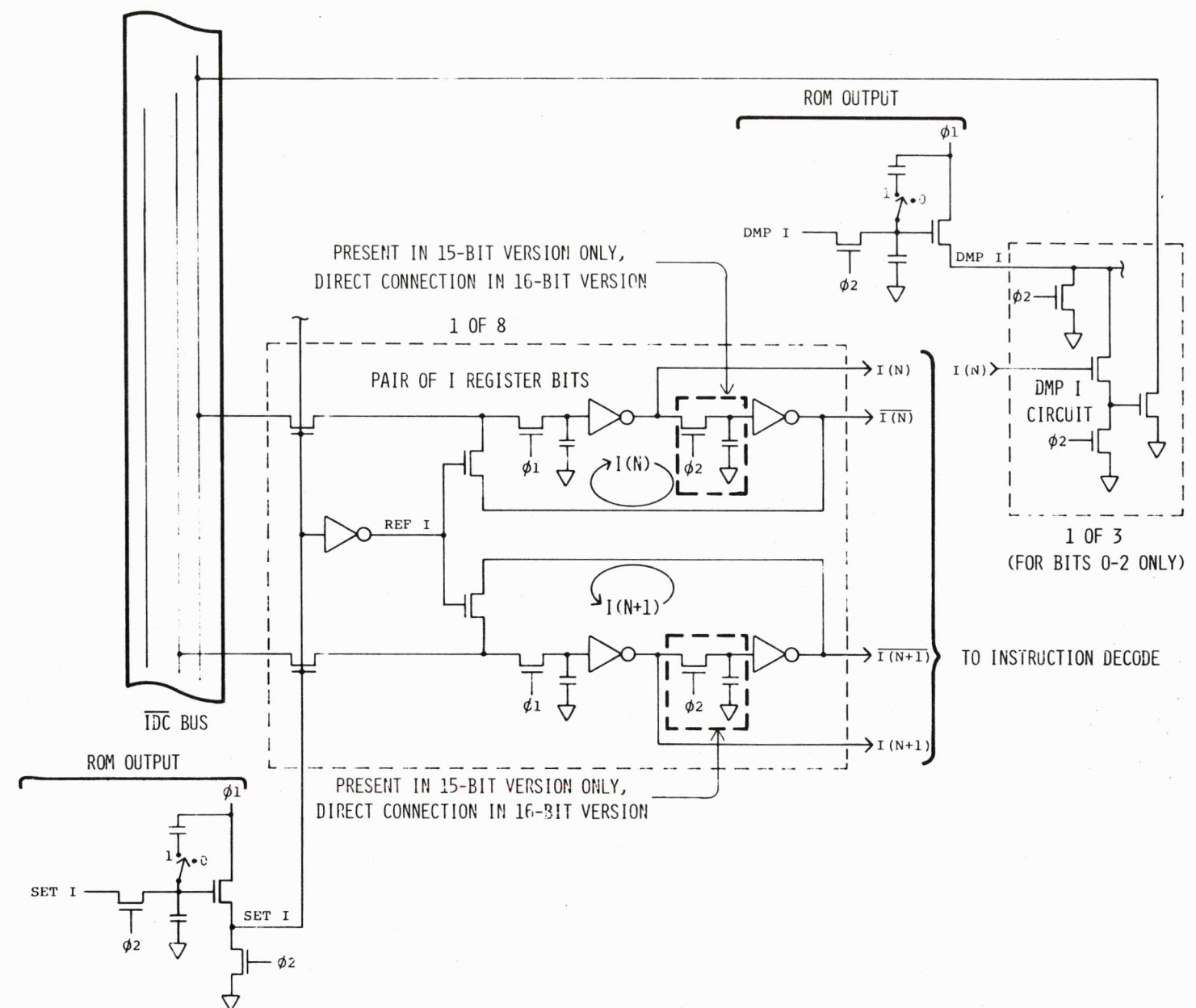
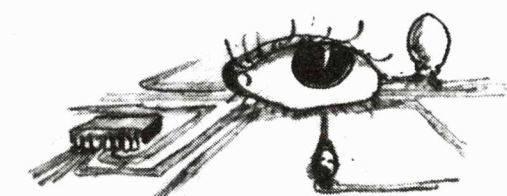


FIG 3-2

Figure 3-2 shows the details of the individual cells of the I register. The difference between the 15 and 16-bit versions is in the way a typical I register cell is constructed. In the 16-bit version the phase two portion of the register cell was eliminated. This results in faster transmission of the instruction bit pattern to Instruction Decode. Such a change is permissible in the case of the I register, since the I register is never dumped at the same time it is being set.



"I" REGISTER

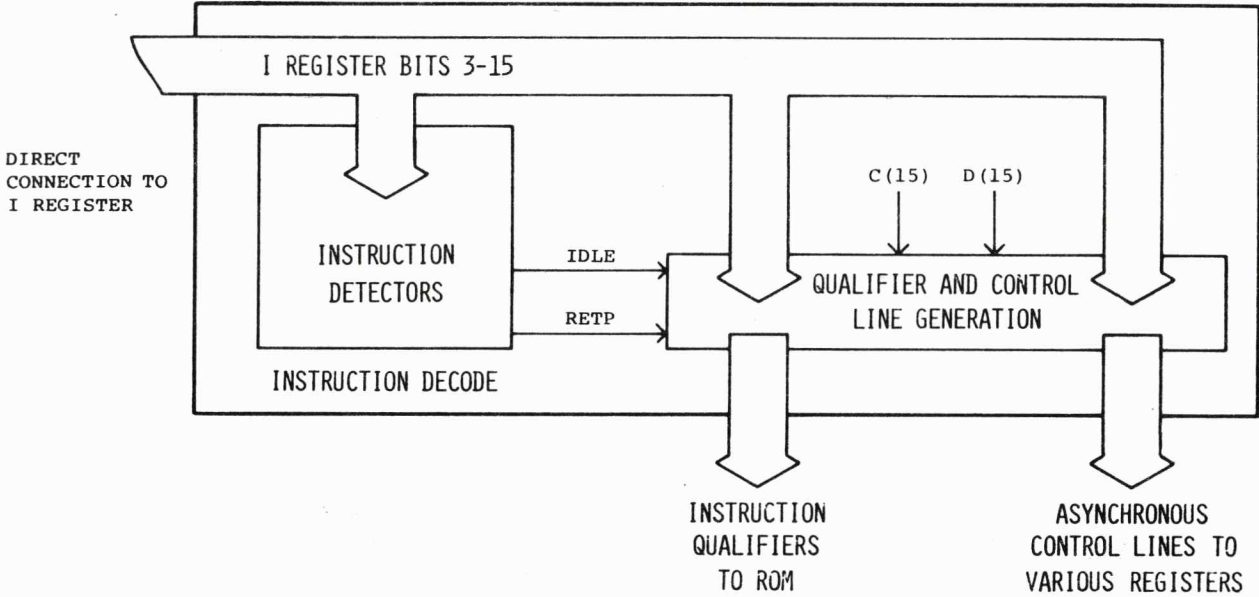


FIG 4-1

TABLE OF INSTRUCTION BIT PATTERNS

GROUP: STACK

INST. NAME	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWC	0	1	1	1	0	0	0	1	I/D	1	1	0	0	* 3 BIT REGISTER ADDRESS FIELD (0 - 7 ₈).		
PBC	0	1	1	1	1	0	0	1	I/D	1	1	0	0			
PWD	0	1	1	1	0	0	0	1	I/D	1	1	0	1			
PBD	0	1	1	1	1	0	0	1	I/D	1	1	0	1			
WWC	0	1	1	1	0	0	0	1	I/D	1	1	1	0			
WBC	0	1	1	1	1	0	0	1	I/D	1	1	1	0			
WWD	0	1	1	1	0	0	0	1	I/D	1	1	1	1			
WBD	0	1	1	1	1	0	0	1	I/D	1	1	1	1			

Annotations: IDLE1 points to bit 11. WD/BY points to bit 10. INC/DEC points to bit 9. PL/WC points to bit 6. C/D points to bit 5.

1. I/D (INCREMENT/DECREMENT) IS ENCODED AS 0/1

GROUP: INTERRUPT

INST. NAME	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIR	0	1	1	1	0	0	0	1	0	0	0	1	0	0	0	0
DIR	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	0

GROUP: DMA(15/16 BIT)

INST. NAME	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA	0	1	1	1	0	0	0	1	0	0	1	0	0	0	0	0
PCM	0	1	1	1	0	0	0	1	0	0	1	0	1	0	0	0
DDR	0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0

GROUP: RETURN

INST. NAME		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RETP {	RET	1	1	1	1	0	0	0	0	1	1	6 BIT, 2'S COMPLEMENT SKIP FIELD (ALLOWS -32 THRU +31).					
		P/P (DON'T POP/POP THE IOC) ENCODED AS 0/1.															

GROUP: 16-BIT ONLY

INST. NAME	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDO	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0
SDI	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0	0
DBL	0	1	1	1	0	0	0	1	0	1	0	0	0	0	0	0
CBL	0	1	1	1	0	0	0	1	0	1	0	0	1	0	0	0
DBU	0	1	1	1	0	0	0	1	0	1	0	1	0	0	0	0
CBU	0	1	1	1	0	0	0	1	0	1	0	1	1	0	0	0

Annotations: IDLE2 points to bit 11. SPECIFIES DMAD points to bit 6. SPECIFIES (C+D) DMAD points to bit 5. UPPER/LOWER BLOCK points to bit 4. IN/OUT points to bit 3. C/D points to bit 2.

FIG 4-2-S

SECTION 4

Figure 4-1 is an overview of Instruction Decode. A collection of instruction detectors monitors the bit pattern in the I register. If the machine-instruction in the I register does not pertain to the IOC, or if it is a RET,P instruction, IDLE is true. A RET,P is represented by RETP. IDLE and RETP are used to enable the generation of qualifiers and asynchronous control lines. The qualifiers serve the same function as the group qualifiers in the BPC. The asynchronous control lines also serve the same function as their counterparts in the BPC.

Figure 4-2 illustrates the various instruction bit patterns for IOC machine-instructions and their relationship with the various signals generated by Instruction Decode.

TABLE OF INSTRUCTION BIT PATTERNS

GROUP: STACK	WD/BY	INC/DEC	REPRESENTS ANY PATTERN EXCEPT 0110, X00X, 10XX
INST. NAME	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
PWC	0 1 1 1 0 0 0 1	I/D 1 1 0 0	* 3 BIT REGISTER ADDRESS FIELD (0 - 7 ₈).
PBC	0 1 1 1 1 0 0 1	I/D 1 1 0 0	
PWD	0 1 1 1 0 0 0 1	I/D 1 1 0 1	
PBD	0 1 1 1 1 0 0 1	I/D 1 1 0 1	
WWC	0 1 1 1 0 0 0 1	I/D 1 1 1 0	
WBC	0 1 1 1 1 0 0 1	I/D 1 1 1 0	
WWD	0 1 1 1 0 0 0 1	I/D 1 1 1 1	
WBD	0 1 1 1 1 0 0 1	I/D 1 1 1 1	

1. I/D (INCREMENT/DECREMENT) IS ENCODED AS 0/1

GROUP: INTERRUPT	INST. NAME	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
EIR		0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 0
DIR		0 1 1 1 0 0 0 1 0 0 0 1 1 0 0 0

GROUP: DMA	INST. NAME	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
DMA		0 1 1 1 0 0 0 1 0 0 1 0 0 0 0 0
PCM		0 1 1 1 0 0 0 1 0 0 1 0 1 0 0 0
DDR		0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0

GROUP: RETURN	INST. NAME	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RETP	RET	1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1

6 BIT, 2'S COMPLEMENT SKIP FIELD (ALLOWS -32 THRU +31).

P/P (DON'T POP/POP THE IOC) ENCODED AS 0/1.

FIG 4-2-F

SECTION 4 (CONTINUED)

Figure 4-3 illustrates the circuitry of the instruction detectors, which generate IDLE and RETP. In the 15-bit version IDLE is a single signal. However, in the 16-bit version it was necessary to separate the notion of IDLE into two parts. The 16-bit version's IDLE1 corresponds pretty much to IDLE in the 15-bit version. IDLE2 corresponds to new machine-instructions that occur only with the 16-bit version. These new instructions concern the CB, DB, and DMAD registers.

DETAILS OF INSTRUCTION DETECTION & u-INSTRUCTION GENERATION FOR THE CB, DB & DMAD REGISTERS.

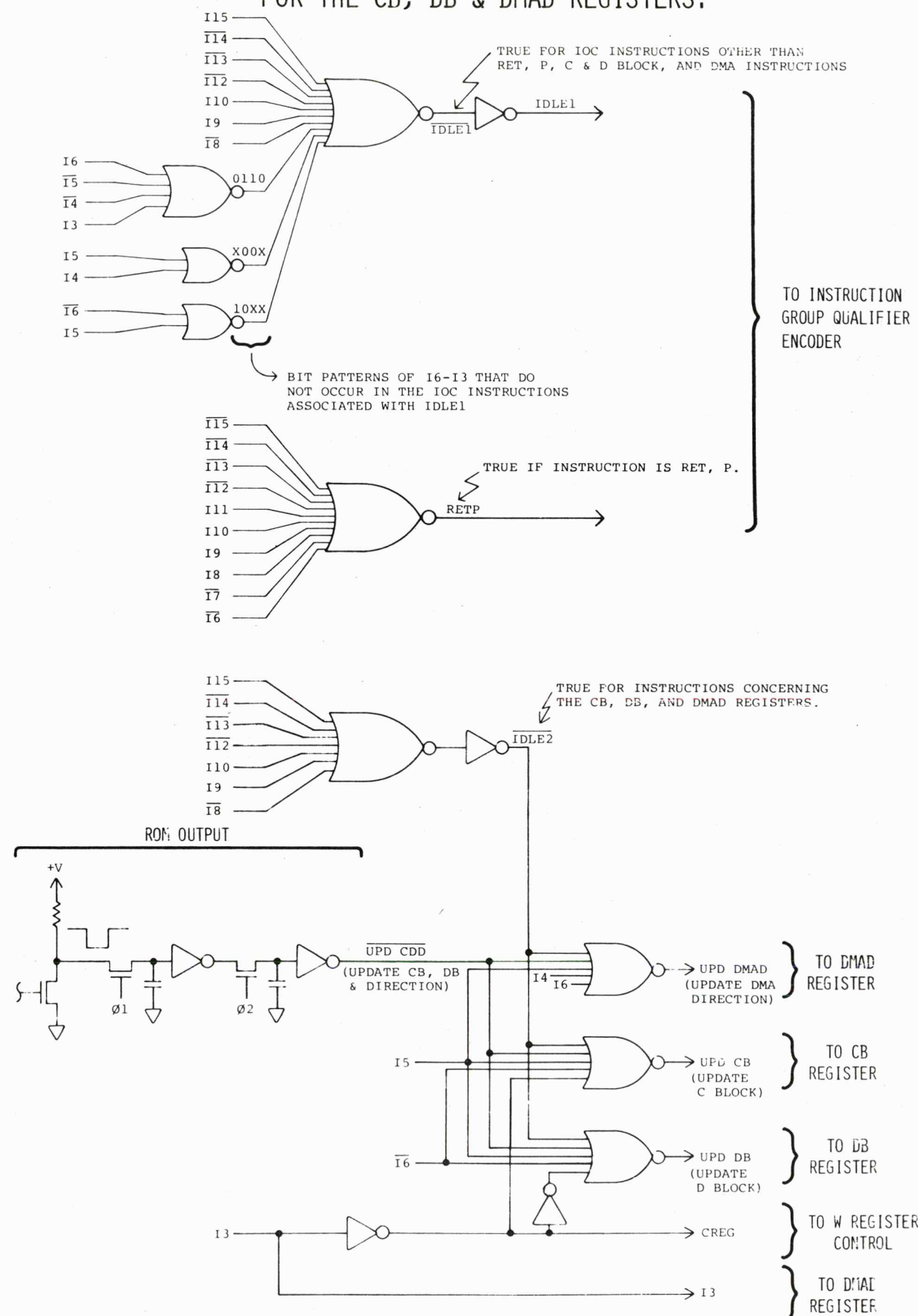


FIG 4-3-S

DETAILS OF INSTRUCTION QUALIFIER AND ASYNCHRONOUS CONTROL LINE GENERATION

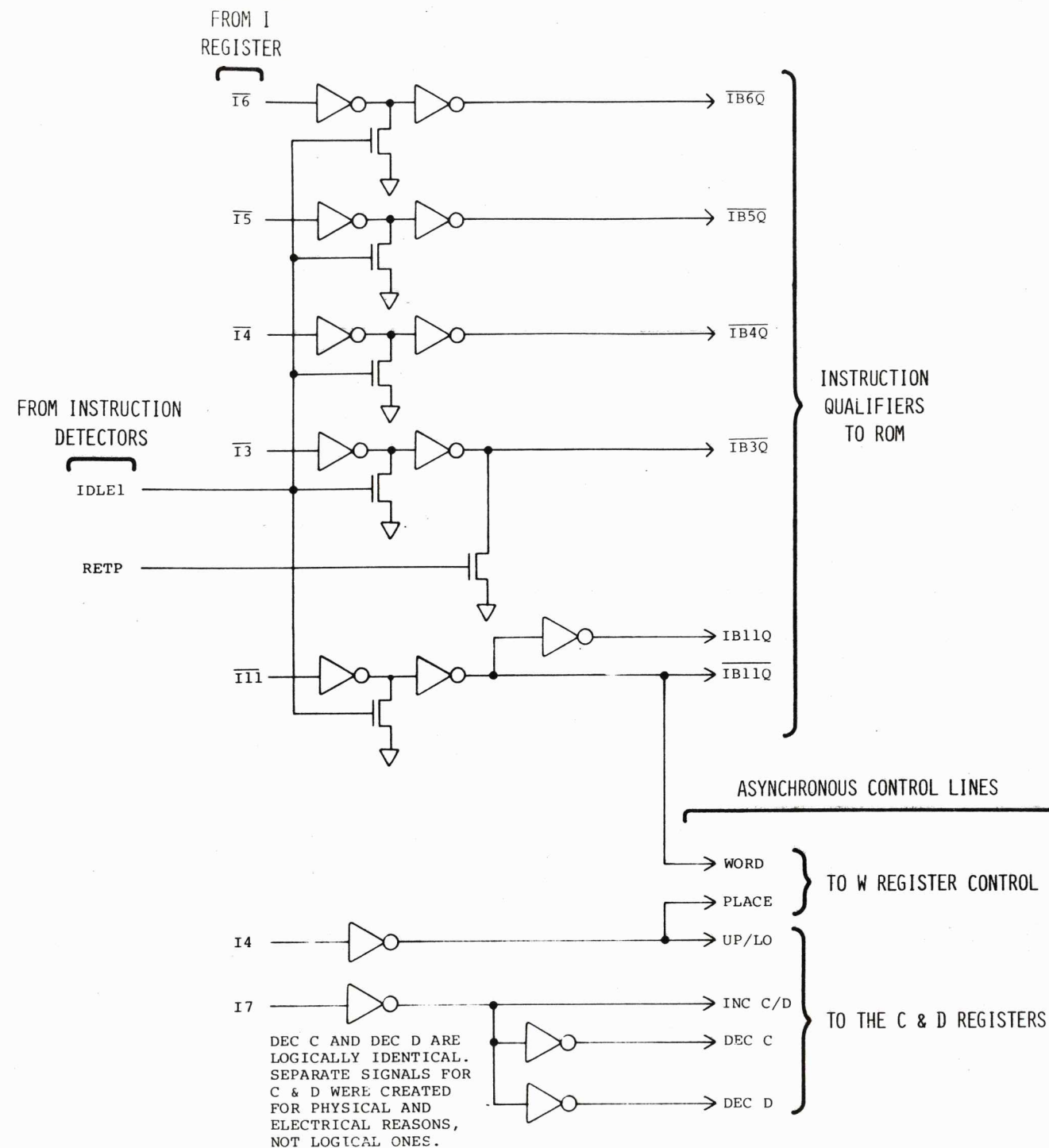


FIG 4-4-S

DETAILS OF INSTRUCTION DETECTION

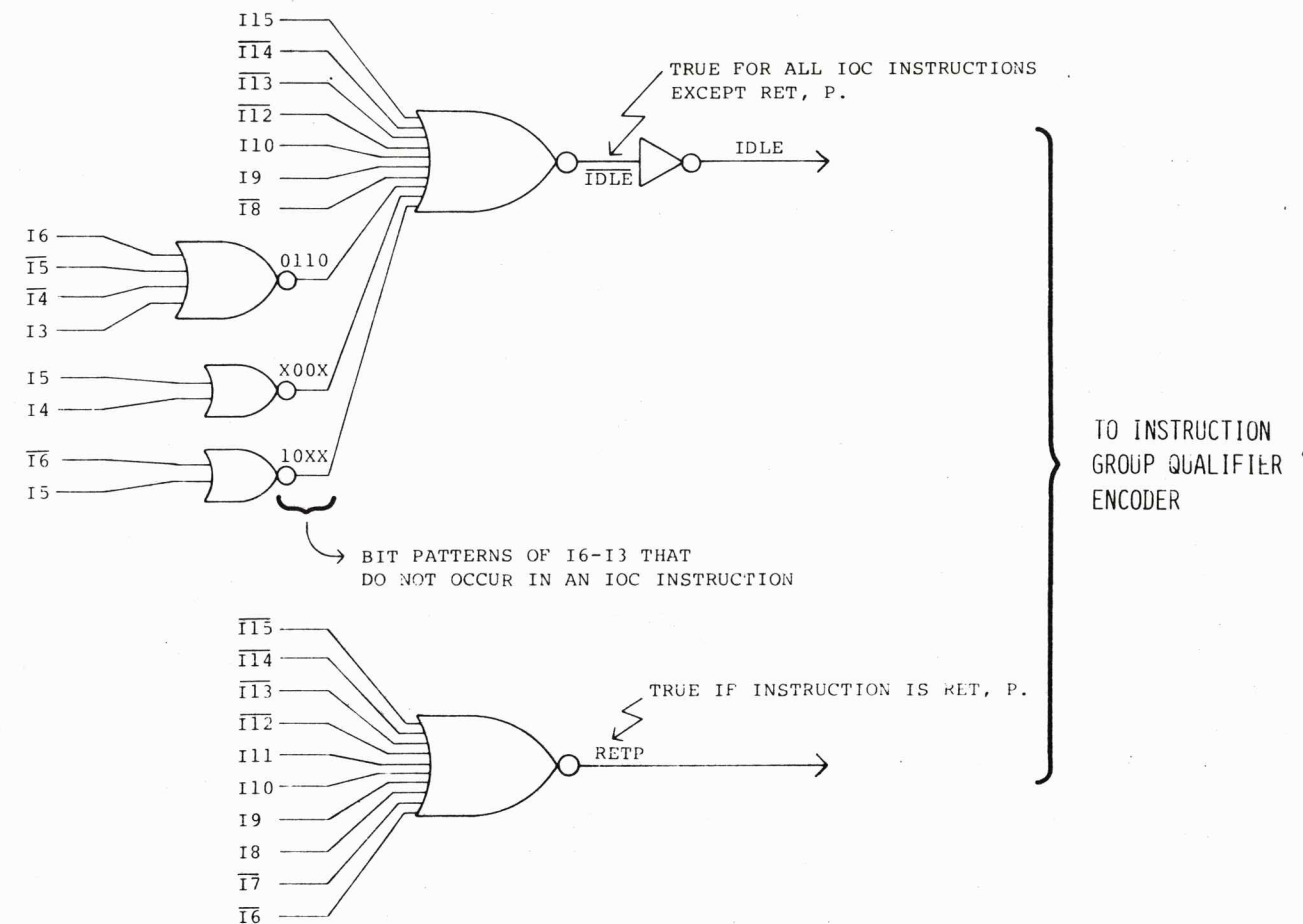


FIG 4-3-F

SECTION 4 (CONTINUED)

Figure 4-4 shows the manner in which the various IDLE signals, RETP and certain other signals are used to generate qualifiers for the ROM and the asynchronous control lines. Significant differences exist in the manner in which the asynchronous control lines relate to the W, C, and D registers. There is no underlying pattern in these differences, other than to say that those registers were implemented somewhat differently between the two versions.

INSTRUCTION
DETECTION

INSTR. QUAL. &
ASYNC. CONT. GEN.

INSTR. DETECTION &
u-INSTR. FOR CB,
DB & DMAD

INSTRUCTION
BIT PATTERNS

DETAILS OF INSTRUCTION QUALIFIER & ASYNCHRONOUS CONTROL LINE GENERATION

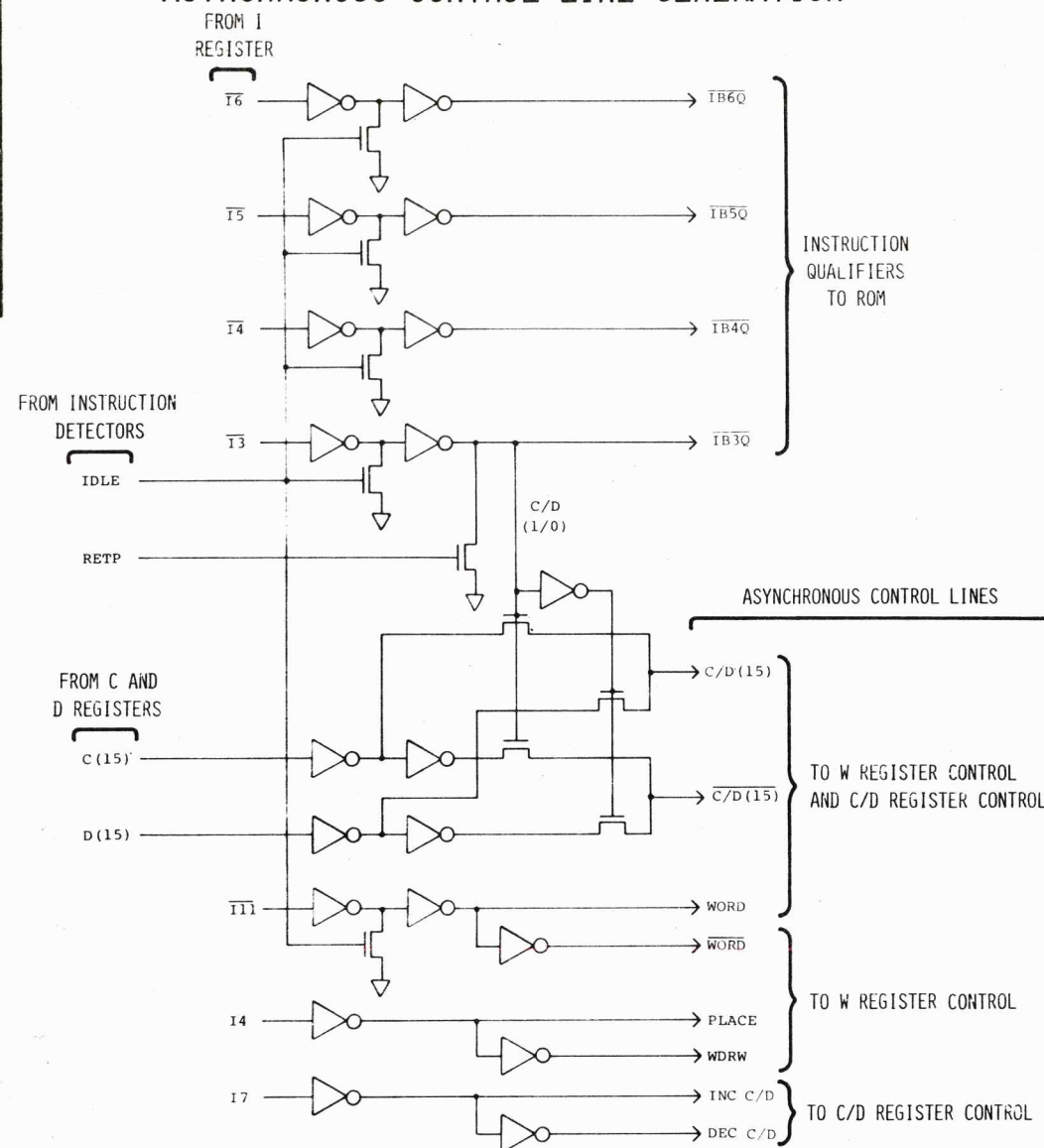


FIG 4-4-F

SECTION 5-S

Figure 5-1-S is an overview of the C, CB, D and DB registers of the 16-bit version. The primary use of these registers is as counters; they are used as pointers during the indirect addressing associated with Place and Withdraw Instructions. Each of the combinations CB-C and DB-D is equipped with an automatic increment/decrement mechanism. Each combination amounts to a 17-bit register, and the Increment/Decrement Network treats the combination as such. Differences between word addresses and byte addresses are obtained by the manner in which the register combinations are dumped to

the IDC Bus, rather than in the way the register is incremented or decremented. This is a significant difference between the 16-bit version and the 15-bit version.

The micro-instructions SET C and SET D are quite straightforward. Values of CB and DB are controlled with three signals; UP/LO used in conjunction with either UPD CB or UPD DB. UP/LO determines whether the affected register is to be set or cleared. UPD CB updates the CB register to match the state of UP/LO. UPD DB performs a similar function for DB. These three signals originate in Instruction Decode. UP/LO

OVERVIEW OF THE C, D, CB AND DB REGISTERS

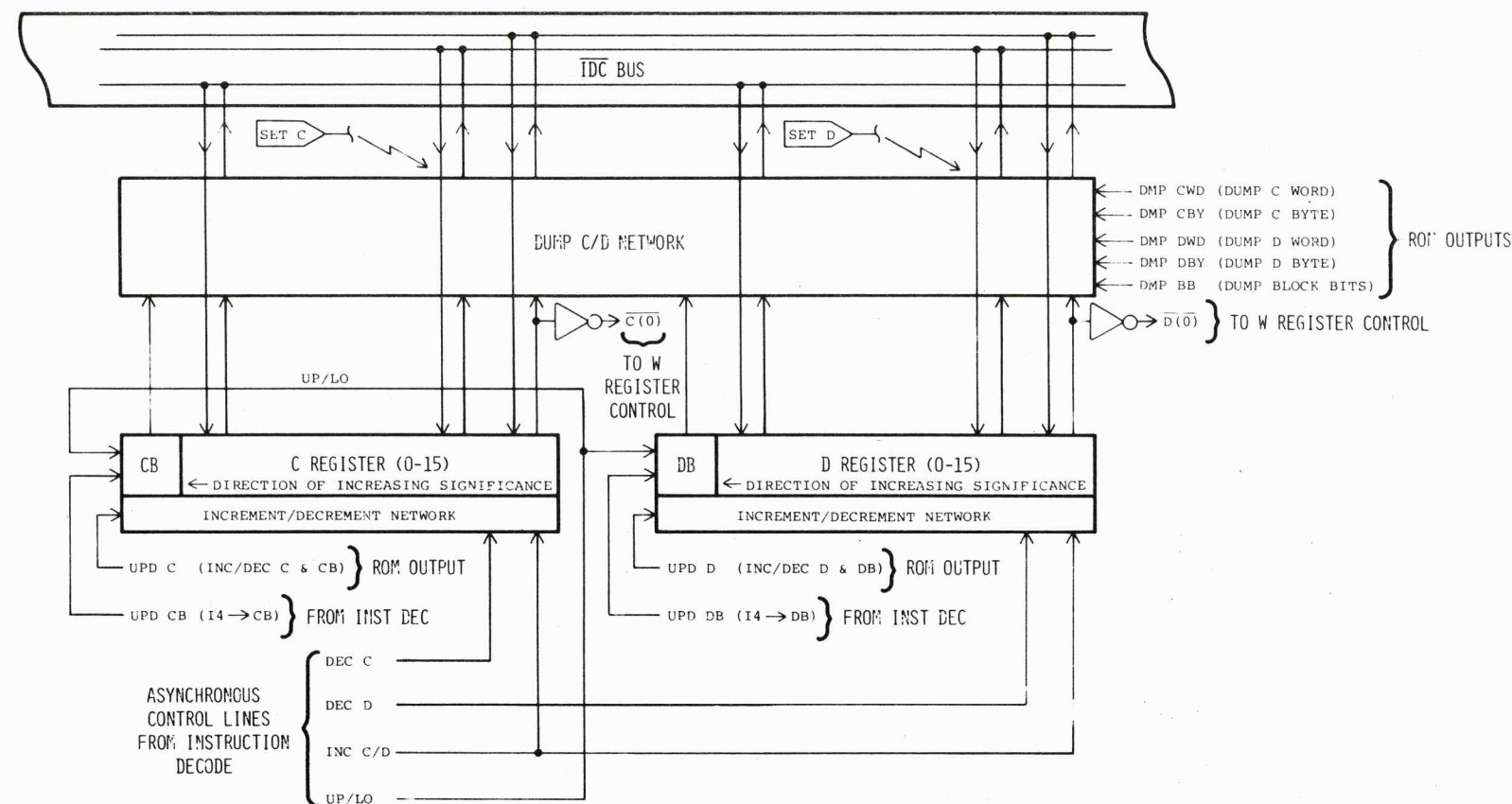


FIG 5-1-S

merely represents bit 4 of the machine-instruction bit pattern. Instruction Decode generates UPD CB and UPD DB from a ROM micro-instruction called UPD CDD. It uses UPD CDD to generate one or the other, based on the bit pattern of the machine-instruction in the I register.

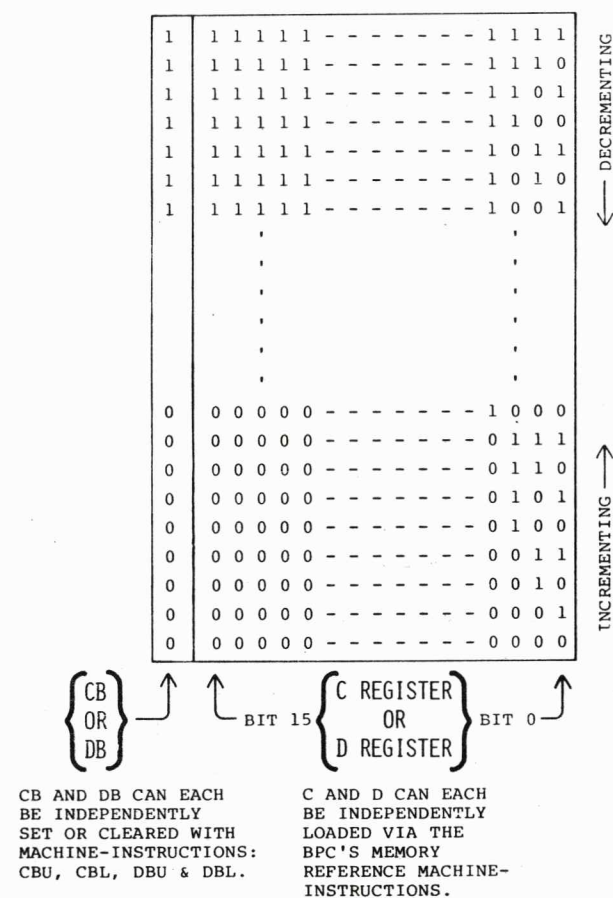
The DMP micro-instructions are complex. To achieve the needed flexibility, a number of DMP micro-instructions were implemented. The names of these instructions and their meanings are shown on the drawing. A desired DMP function is created when the ROM decodes the necessary combination of DMP micro-instructions. The DMP C/D Network implements these micro-instructions. The main difference between the micro-instructions is in the way in which the various bits of the C and D registers are mapped onto the IDC Bus.

The incrementing and decrementing of the 17-bit register combinations is done as follows. First, Instruction Decode generates

signals to determine whether to increment or decrement when the command to update is given. A single signal represents the selection of the increment mode to both registers. That signal is INC C/D. A pair of signals represents the decrement mode; these are DEC C and DEC D. These latter two signals could have been a single signal going to both registers. As it is, they are logically identical and are separate signals only for lay-out and electrical reasons. The ROM decodes separate commands to update each 17-bit register combination. These micro-instructions are UPD C and UPD D. Each 17-bit combination will increment or decrement one count upon receipt of its update command.

Observe that C(0) and D(0) are sent to W register control. These are used in implementing cross-over operations during byte oriented Place and Withdraw machine-instructions.

C-CB AND D-DB REGISTER INCREMENTING AND DECREMENTING SEQUENCES, AND HOW THEY ARE DUMPED TO THE BPC



TO INCREMENT:
STARTING WITH BIT 0, COMPLEMENT EACH SUCCESSIVE LEFT-WARD BIT UNTIL A ZERO HAS BEEN COMPLEMENTED TO A ONE.

TO DECREMENT:
STARTING WITH BIT 0, COMPLEMENT EACH SUCCESSIVE LEFT-WARD BIT UNTIL A ONE HAS BEEN COMPLEMENTED TO A ZERO.

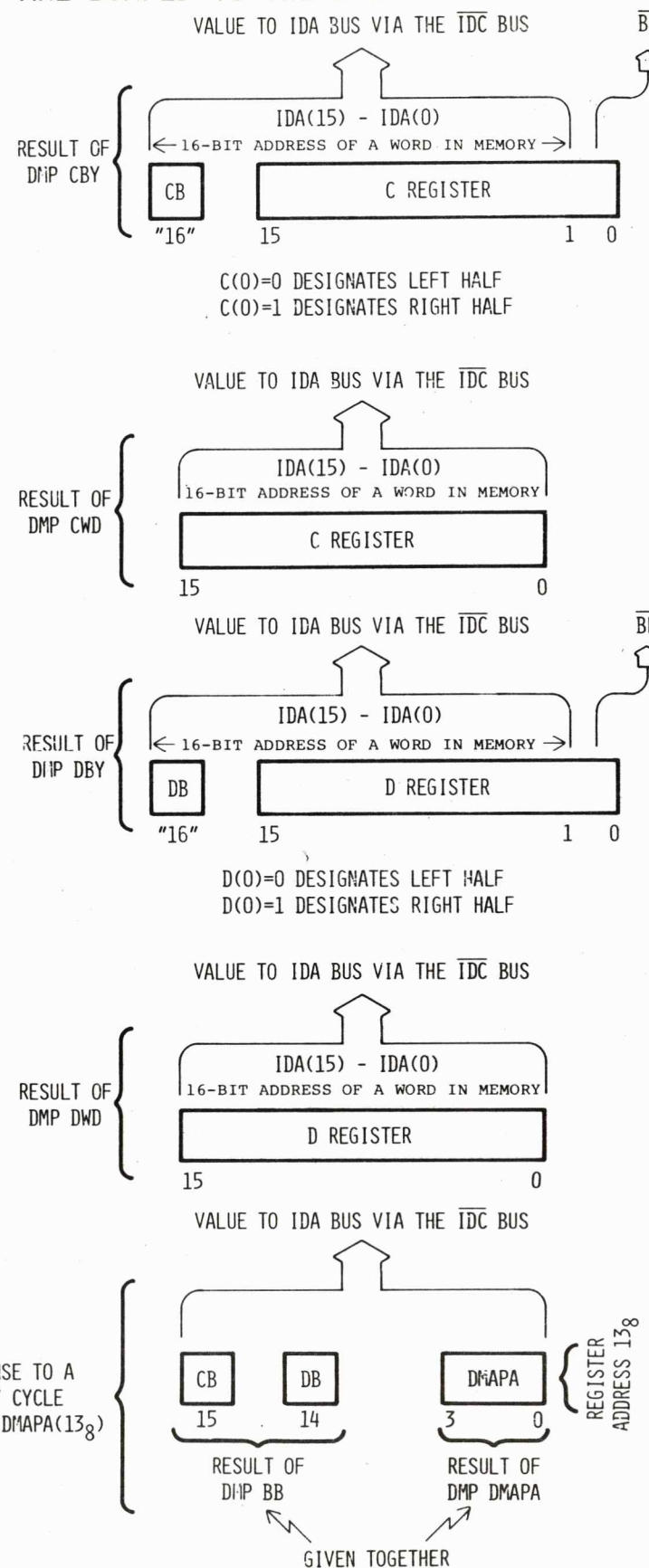


FIG 5-2-S

P/O THE DUMP C/D NETWORK-GENERATION OF THE DMP SIGNALS

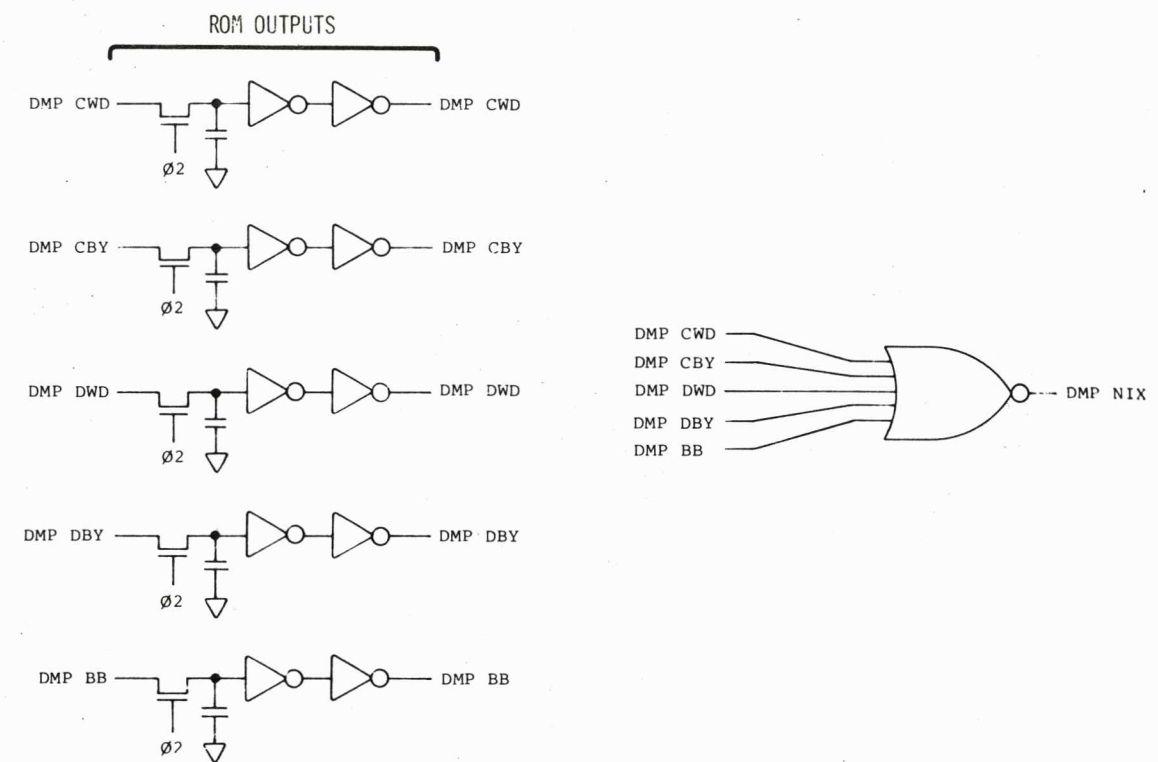


FIG 5-3-I-S

SECTION 5-S (CONTINUED)

Figure 5-2-S illustrates the incrementing and decrementing operations for both word and byte addressing associated with the C and D registers. Observe how CB and DB are employed during byte addressing operations, and how the associated special use of bit 0 of the C and D registers generates BL. The figure also illustrates the various DMP signals necessary to generate proper addressing under the various possible conditions.

Figures 5-3-S illustrate the Dump C/D Network. Observe how the NOR of the various DMP signals is used to enable the pull-down of the various IDC Bus lines. Also observe that the various DMP micro-instructions put different numbered bits onto the IDC Bus lines.

DMP BB provides a special dump of CB and DB onto the IDC Bus. DMP BB is issued as part of a response to a read memory cycle directed to register 13₈ (DMAPA). This is provided simply as a means to allow interrogation of the values of CB and DB.

P/O DUMP C/D NETWORK

C-CB AND D-DB INC. AND DEC. SEQUENCES

C, D, CB AND DB OVERVIEW

INSTR. QUAL. & ASYNC. CONT. GEN.

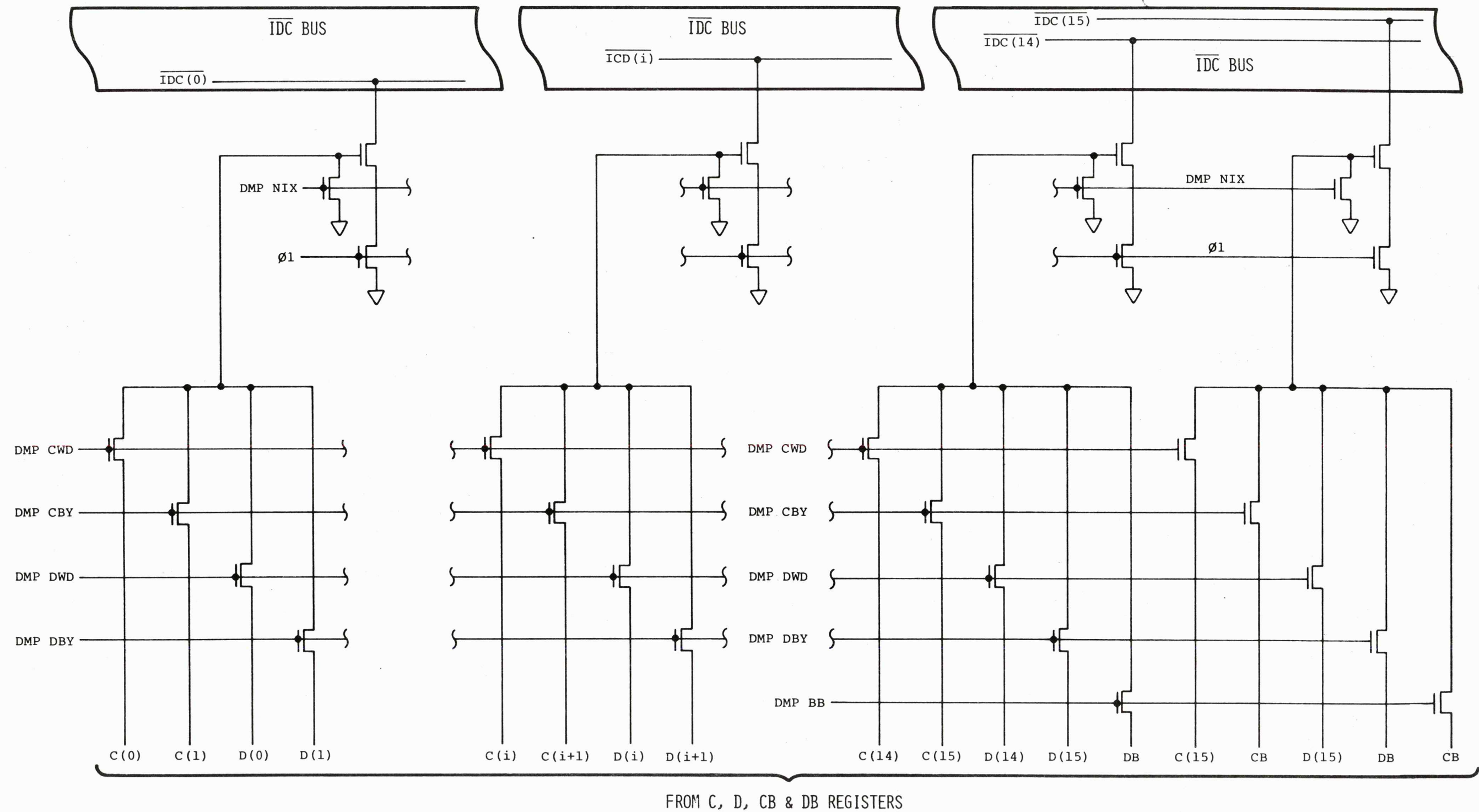
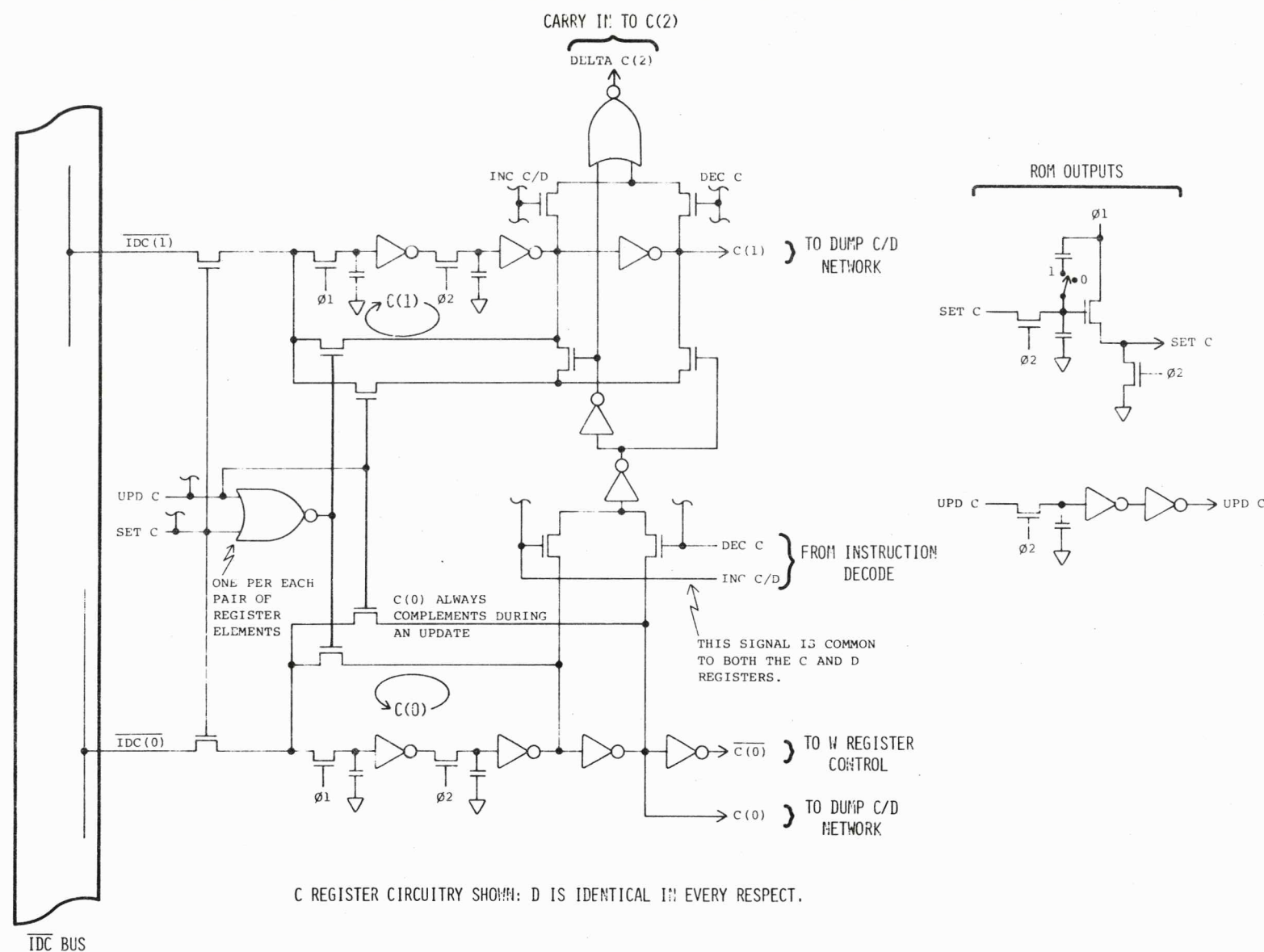


FIG 5-3-2-S

DETAILS OF C AND D REGISTERS, BITS 0 & 1, SET, AND COUNT CIRCUITS



C REGISTER CIRCUITRY SHOWN: D IS IDENTICAL IN EVERY RESPECT.

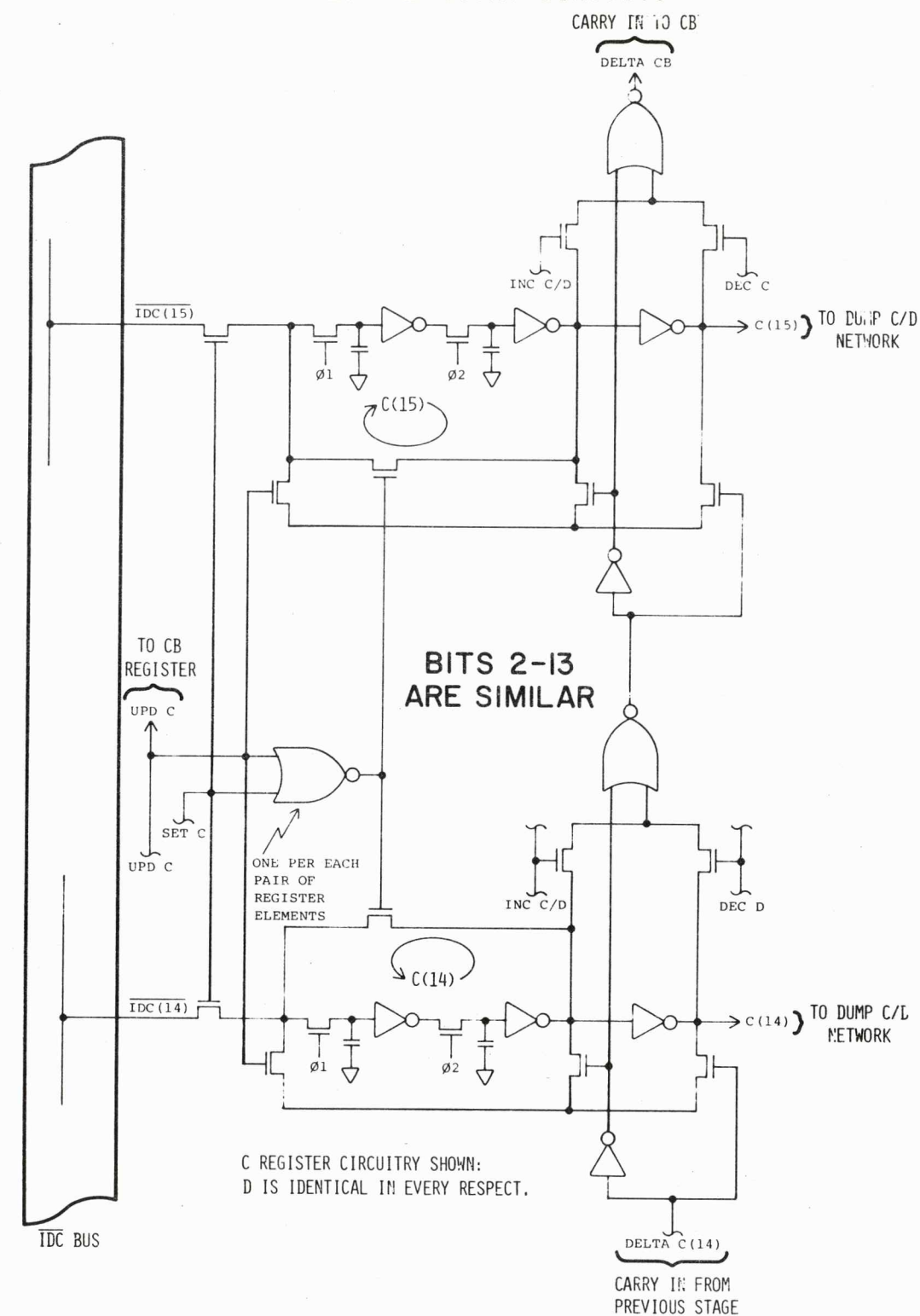
FIG 5-4-S

SECTION 5-S (CONTINUED)

Figure 5-4-S shows the details of the first two bits of the C and D registers in the 16-bit version. Note that C(0) *always* complements during an update.

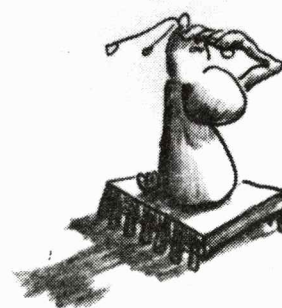
Figure 5-5-S shows the details of the remaining bits of the C and D registers.

DETAILS OF C AND D REGISTERS, BITS 2-15, SET AND COUNT CIRCUITS



C REGISTER CIRCUITRY SHOWN: D IS IDENTICAL IN EVERY RESPECT.

FIG 5-5-S



THE "SET" REGISTER

S

P/O C AND D
REGISTERS

S

P/O C AND D
REGISTERS

S

P/O THE DUMP C/D

DETAILS OF THE CB AND DB REGISTERS

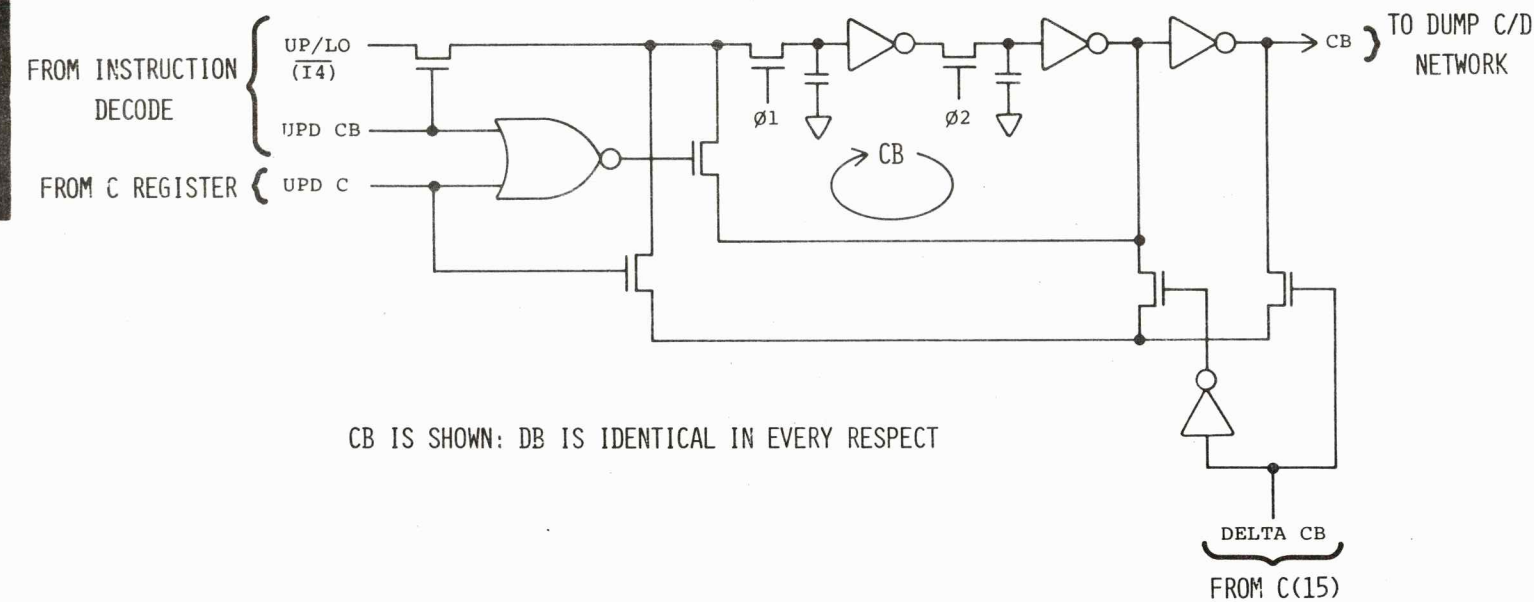


FIG 5-6-S

OVERVIEW OF THE C AND D REGISTERS

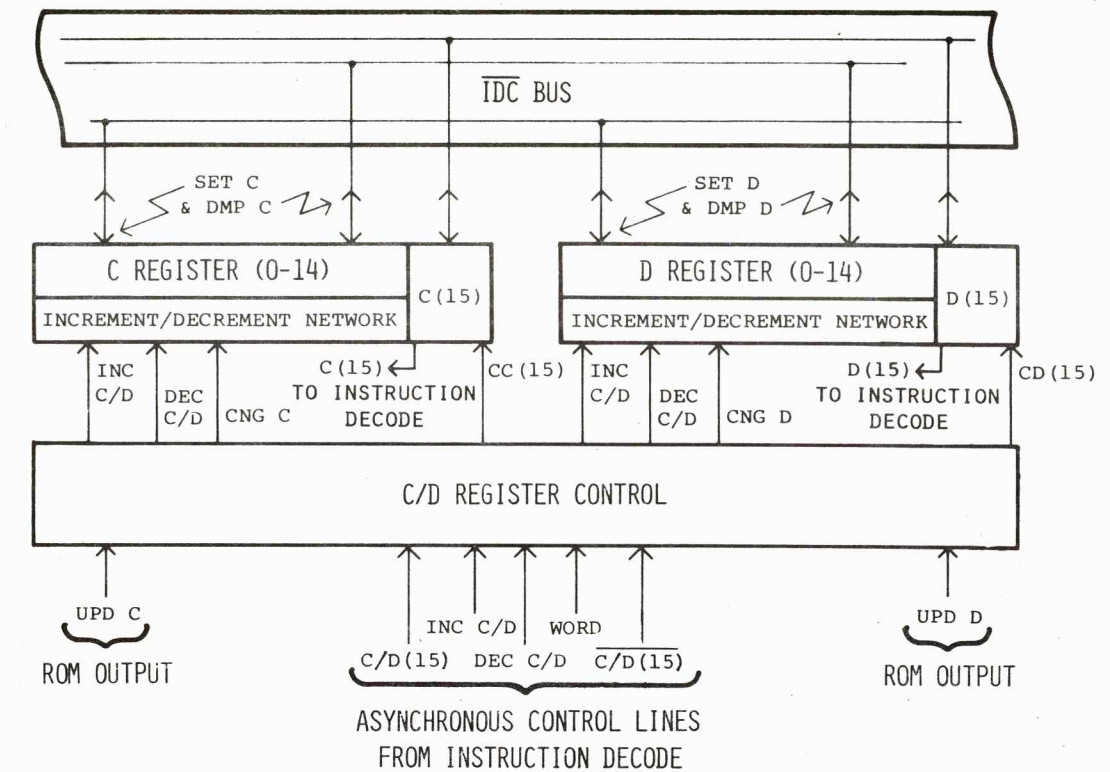


FIG 5-1-F

SECTION 5-S (CONTINUED)

Figure 5-6-S illustrates the circuitry of the CB and DB registers. Observe that they are connected to their respective Increment/Decrement Networks as if they were a 17th bit of their associated register.

SECTION 5-F

Figure 5-1-F is an overview of the C and D registers, as employed in the 15-bit version. Observe that the Increment/Decrement Network serves only the lower 15 bits of each register.

The micro-instructions SET C and SET D are quite straightforward, as are DMP C and DMP D. However, there is considerable funny-business concerning the manner in which these registers are incremented and decremented during byte oriented operations.

During word oriented operations the increment and decrement is a straight binary operation on the least-significant 15 bits of the associated register. The 16th bit is not involved. C/D Register Control uses the signals INC C/D, DEC C/D and WORD (all of which are from Instruction Decode), in conjunction with the ROM outputs UPD C (Update C) and UPD D (Update D), to generate control signals to each register. These signals are INC C/D and DEC C/D (which go to both C and D), and CNG C and CNG D. The

latter two signals are the actual instructions to perform the increment or decrement, and are derived from the update micro-instruction for the associated register. During word operations each update micro-instruction *always* results in an associated change signal.

The situation is somewhat different during byte oriented operations. During byte operations C/D(15) is adjusted by Instruction Decode to represent bit 15 of the register to be incremented or decremented. This signal will be used to decide whether or not to increment or decrement the lower 15 bits of the register being altered. Furthermore, C/D Register Control now arranges that each update automatically results in a complement instruction to bit 15 of the associated register. Whether or not the change (CNG) signal is issued depends upon the previous value of that bit, as reflected by C/D(15). Thus, the 16th bit *always* toggles, and the remaining 15 bits increment or decrement

every other update operation.

Observe that in the 15-bit version C(15) and D(15) are sent to Instruction Decode. It is there that C/D(15) is generated. That signal goes not only to C/D Register Control, but also to W Register Control. It is used by W Register Control in implementing the cross-over function required during byte oriented Place and Withdraw operations.

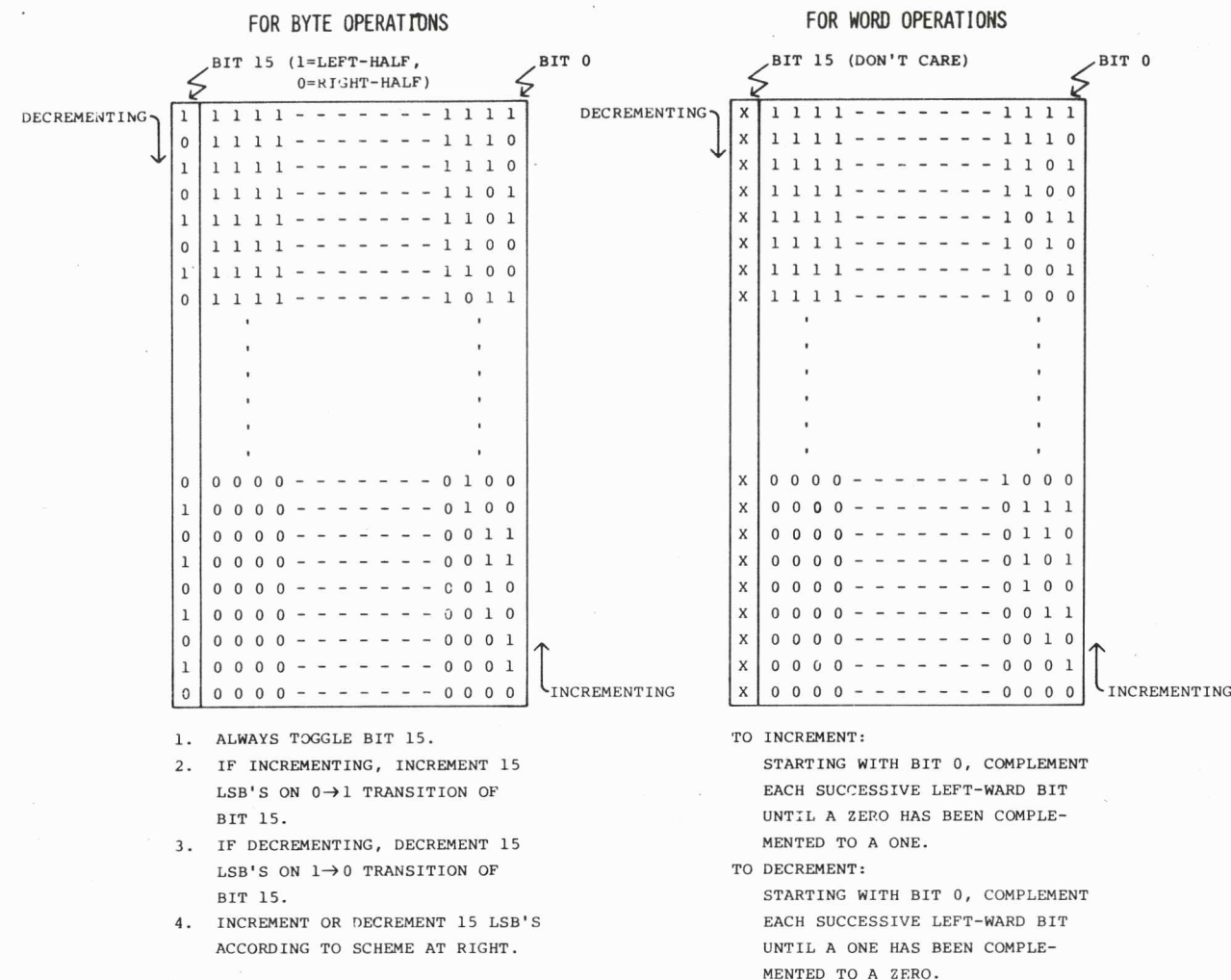


FIG 5-2-F

SECTION 5-F (CONTINUED)

Figure 5-2-F illustrates the incrementing and decrementing operations for both word and byte addressing, as associated with the C and D registers. Observe the odd-ball incrementing and decrementing pattern used during byte operations.

Figure 5-3-F illustrates the details of the C/D Register Control circuitry. This circuitry employs a descending hierarchy of control signals. At the highest level of abstraction are the signals UPD C and UPD D. These stand for Update C and Update D, respectively. In this application "Update" merely means to do something to the associated register. In the event that a byte oriented Place or Withdraw machine-instruction is being

executed, the update signal is sufficient information to generate the signals CC(15) and CD(15). These signals are used to toggle bit 15 of the associated register. An additional pair of signals are derived from the update signals. These are CNG C and CNG D. They are the actual commands to increment or decrement the associated register. If word addressing is in effect every "update" is accompanied by a "change". During byte addressing, however, whether or not to "change" the register depends upon the state of bit 15 of that register, and also upon whether or not the register is being incremented or decremented. Lastly, Instruction Decode provides a pair of lines called INC C/D and DEC C/D. These signals affect the count circuitry within the registers

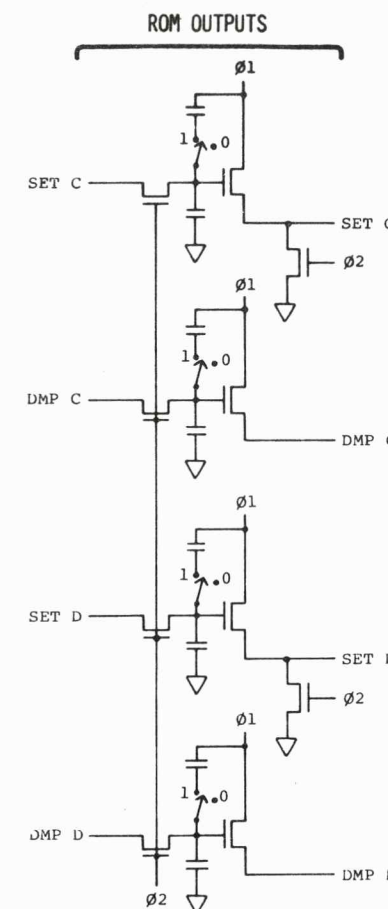
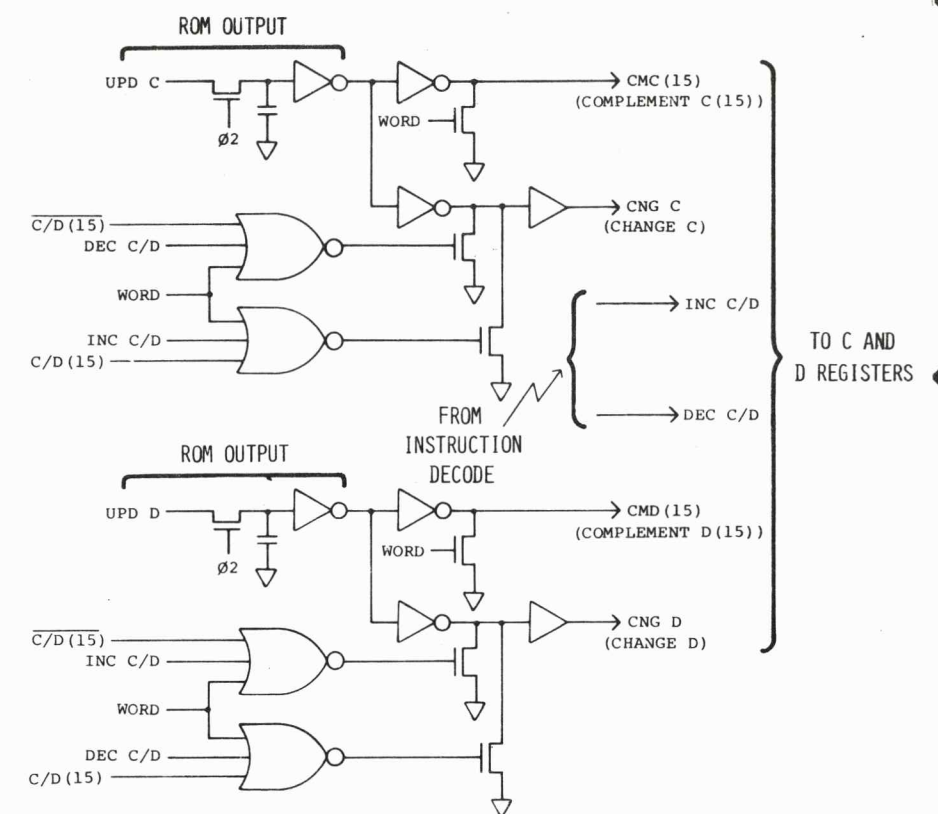


FIG 5-3-F

themselves, to determine whether a "change" results in either an increment or a decrement.



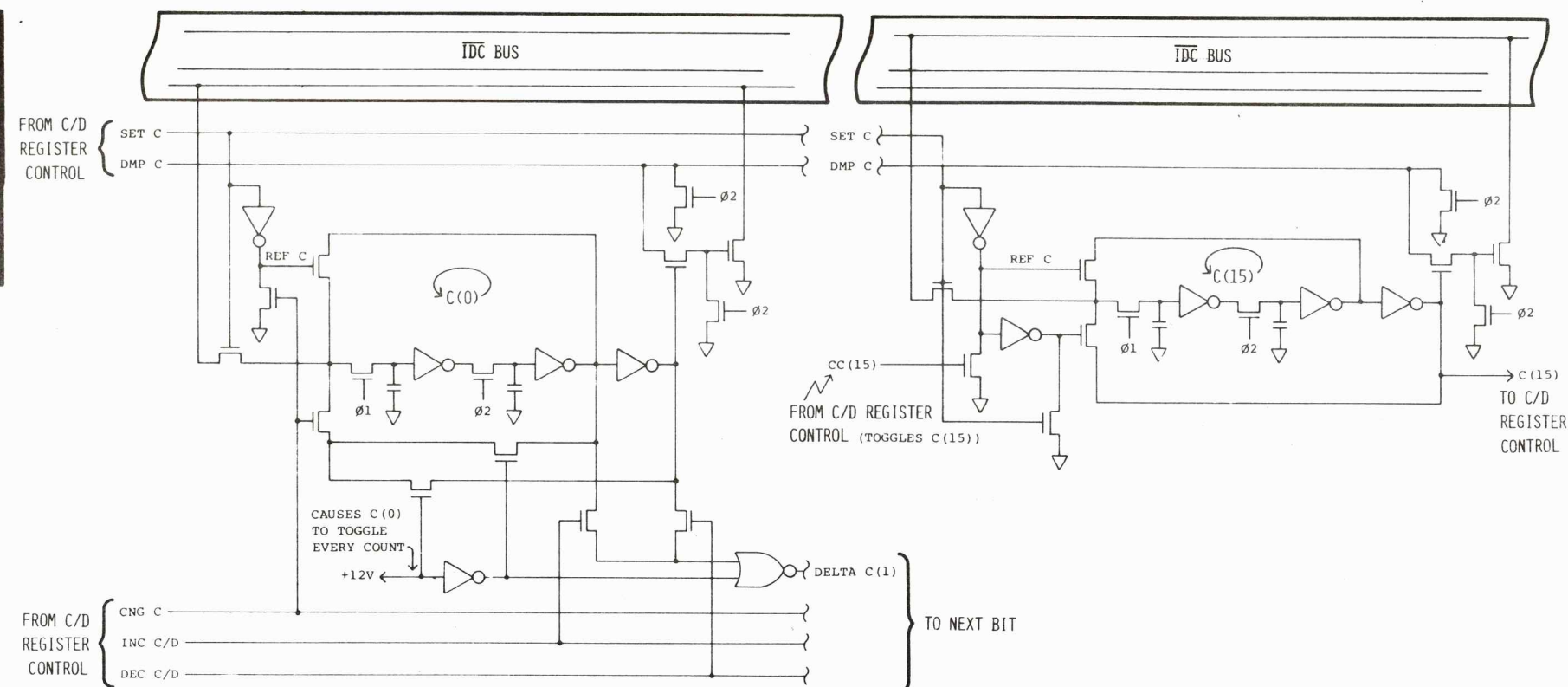
C/D REGISTER CONTROL

C AND D INC. AND DEC. SEQUENCES

C AND D OVERVIEW

CB AND DB REGISTERS

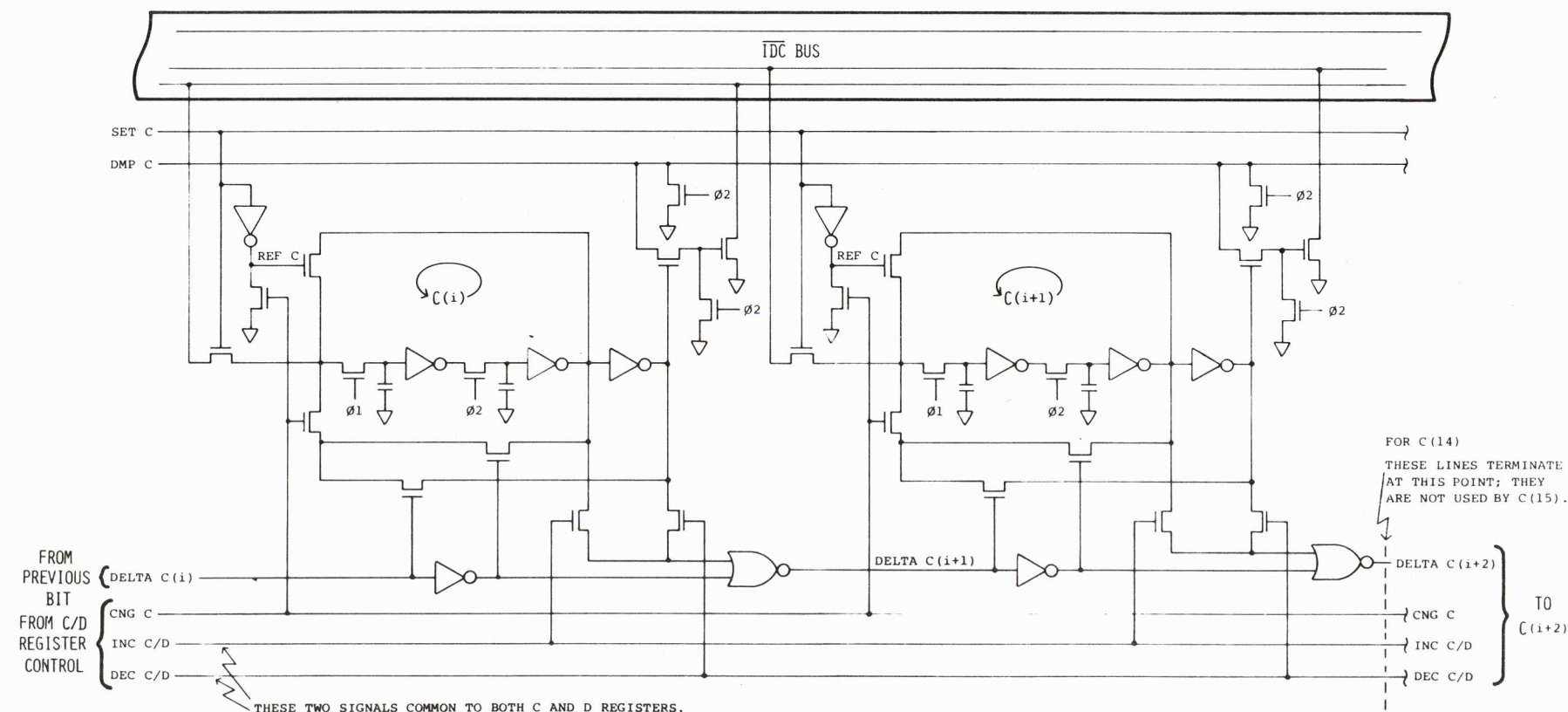
DETAILS OF C AND D REGISTERS, BITS 0 AND 15: SET, DMP AND COUNT CIRCUITS



C REGISTER CIRCUITRY SHOWN: D IS IDENTICAL IN EVERY RESPECT.

FIG 5-4-F

DETAILS OF THE C AND D REGISTERS, BITS 1-14: SET, DMP AND COUNT CIRCUITS.



C REGISTER CIRCUITRY SHOWN: D IS IDENTICAL IN EVERY RESPECT.

FIG 5-5-F

SECTION 5-F (CONTINUED)

Figure 5-4-F illustrates bits 0 and 15 of the C and D registers.

Figure 5-5-F illustrates the remaining bits of the C and D registers.

OVERVIEW OF THE W REGISTER, & ASSOCIATED SET, DUMP, CROSSOVER, AND INTERRUPT VECTOR NETWORKS

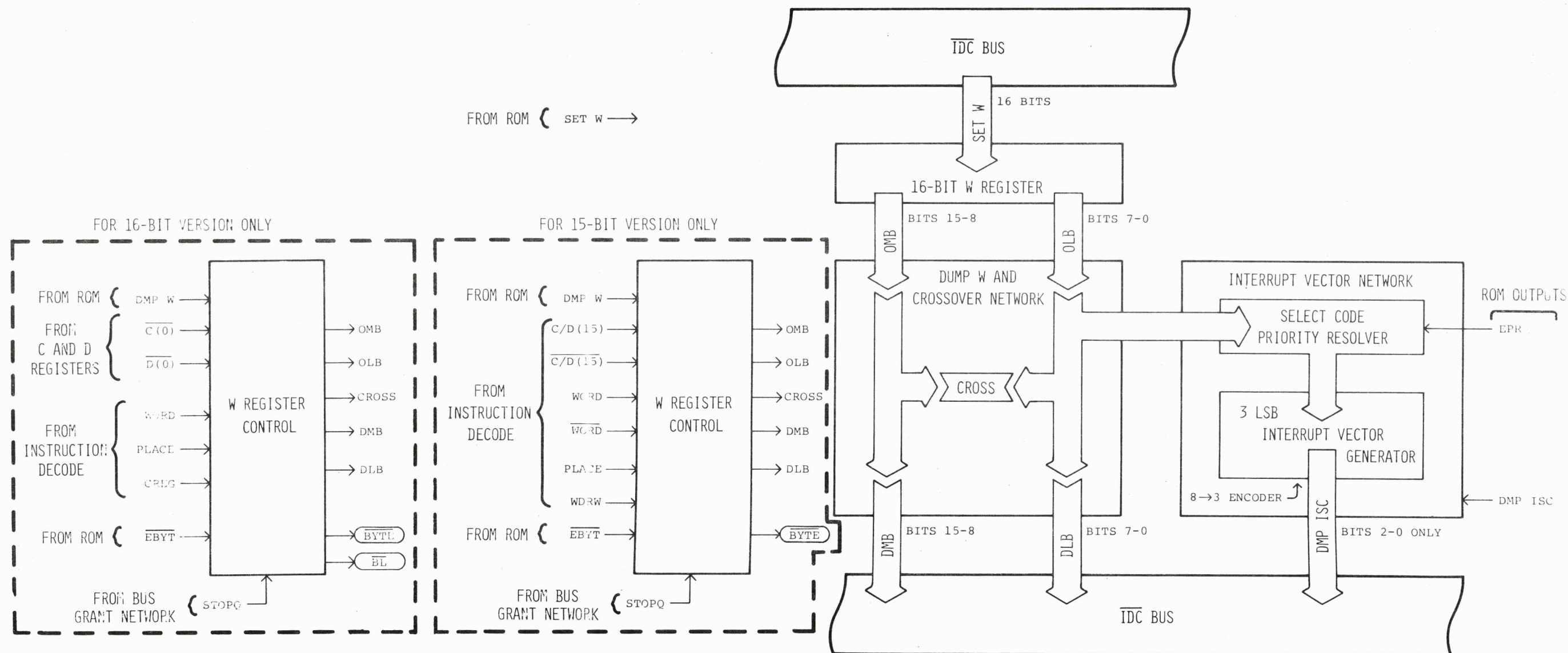


FIG 6-1

SECTION 6

Figure 6-1 is an overview of the W register, W Register Control, DMP W and Crossover Network, and of the Interrupt Vector Network. The 15 and 16-bit versions are similar except for minute differences in implementation concerning W Register Control.

The micro-instruction SET W is quite straightforward. There is, however, a fair amount of complexity surrounding the DMP W micro-instruction. W Register Control resolves DMP W into a combination of other instructions. OMB and OLB are steady-state signals that are generated whenever it is desired to output the most significant 8 bits, or, least significant 8 bits, respectively,

of the W register. CROSS is generated whenever a byte oriented operation is in progress. It is generated even if the transmitted byte is not to change its relative position in the word. That is, even when an upper byte is to remain an upper byte, or a lower byte is to remain a lower byte. The signals DMB and DLB are derived directly from DMP W, in conjunction with some gating in W Register Control. DMB places information onto the upper 8 bits of the IDC Bus. DLB places information on the lower 8 bits of the Bus.

A word oriented DMP W is performed as follows. OMB and OLB will both be true. CROSS will be false. The DMP W issued from the

ROM will generate both DMB and DLB. The net result is a complete 16-bit dump of the W register, in the expected manner.

A byte oriented DMP of W might be performed as follows. Suppose the bottom half of the W register were to be dumped to the upper half of the IDC Bus. In such a case, OLB would be true and OMB false. CROSS would be true but DMP W would generate only DMB and not DLB.

As part of granting an interrupt request the IOC performs an interrupt poll. As a result of the poll the lower 8 bits of the W register contain a record of which peripherals are requesting interrupt

on the level currently in use. It is necessary to be able to identify the highest priority select code that is requesting interrupt, and to generate a binary code that corresponds to its select code number. This is the function of the Interrupt Vector Network.

The Interrupt Vector Network comprises a Select Code Priority Resolver and an Interrupt Vector Generator for encoding the least 3 significant bits of the interrupt vector. In conjunction with OLB and the micro-instruction EPR, the Select Code Priority Resolver identifies the highest number bit which is set among the least 8 significant bits of the W register. That is, it

W REGISTER &
ASSOCIATED CIRCUITS
OVERVIEW

P/O C AND D
REGISTERS

P/O C AND D
REGISTERS

TABLE OF CROSSOVER & NON-CROSSOVER W REGISTER OPERATION

PLACE WORD		WRITING THE ENTIRE WORD OF W	
(REG. 0-7) → (C OR D INDIRECT) INCREMENT/DECREMENT C OR D BEFORE THE ACTUAL PLACE OPERATION	1. START MEMORY CYCLE TO READ SOURCE REGISTER, PUT ENTIRE DATA WORD INTO W. 2. UPDATE C OR D (BEFORE (3)). 3. START MEMORY CYCLE TO WRITE ENTIRE WORD OF W TO MEMORY AT LOCATION C OR D INDIRECT.	<div><div>W REGISTER</div><div>MSB LSB</div><div>↓ ↓</div><div>OMB OLB</div><div>↓ ↓</div><div>DMB DLB</div><div>↓ ↓</div><div>IDC BUS</div></div> <div>OMB=WORD+(WDRW•C/D(0)) OLB=WORD+PLACE+C/D(0) DMB=WORD+(PLACE•C/D(0)) DLB=WORD+WDRW+C/D(0) CROSS=WORD</div> <div>C/D(0) REPRESENT EITHER C(0) OR D(0), WHICHEVER IS APPROPRIATE</div>	
WITHDRAW WORD			
(C OR D INDIRECT) → (REG. 0-7) INCREMENT/DECREMENT C OR D AFTER THE ACTUAL WITHDRAW OPERATION	1. START MEMORY CYCLE TO READ C OR D INDIRECT, PUT ENTIRE DATA WORD INTO W. 2. UPDATE C OR D (AFTER (1)). 3. START MEMORY CYCLE TO WRITE ENTIRE WORD OF W INTO DESTINATION REGISTER.		
PLACE BYTE			
(RIGHT-HAND BYTE OF REG.0-7) → (LEFT OR RIGHT HALF OF C OR D INDIRECT) C/D(0)=0⇒ LEFT HALF C/D(0)=1⇒ RIGHT HALF INCREMENT/DECREMENT C OR D BEFORE THE ACTUAL PLACE OPERATION	1. START MEMORY CYCLE TO READ REGISTER, PUT ENTIRE DATA WORD INTO W. 2. UPDATE C OR D (BEFORE (3)). 3. START A MEMORY CYCLE TO WRITE TO C OR D INDIRECT. 4. AS DATA, OUTPUT THE RIGHT HALF OF W TO EITHER THE UPPER OR LOWER HALF OF THE IDC BUS, AS DETERMINED BY C/D(0).	DESTINATION IS LEFT HALF C/D(0) IS = 0 <div><div>W REGISTER</div><div>MSB LSB</div><div>↓ ↓</div><div>OMB OLB</div><div>↓ ↓</div><div>CROSS</div><div>↓ ↓</div><div>DMB DLB</div><div>↓ ↓</div><div>IDC BUS</div></div>	DESTINATION IS RIGHT HALF C/D(0) IS = 1 <div><div>W REGISTER</div><div>MSB LSB</div><div>↓ ↓</div><div>OMB OLB</div><div>↓ ↓</div><div>CROSS</div><div>↓ ↓</div><div>DMB DLB</div><div>↓ ↓</div><div>IDC BUS</div></div>
WITHDRAW BYTE			
(LEFT OR RIGHT HALF OF C OR D INDIRECT) → (RIGHT HAND BYTE OF REG. 0-7) C/D(0)=0⇒ LEFT HALF C/D(0)=1⇒ RIGHT HALF INCREMENT/DECREMENT C OR D AFTER THE ACTUAL WITHDRAW OPERATION	1. START MEMORY CYCLE TO READ C OR D INDIRECT, PUT ENTIRE DATA WORD INTO W. 2. UPDATE C OR D (BEFORE (3), BUT!!). 3. C/D(0) IS NOW THE OPPOSITE OF WHAT IT WAS AT (1). 4. START A MEMORY CYCLE TO THE DESTINATION REGISTER. 5. AS DATA, OUTPUT TO THE LOWER HALF OF THE IDC BUS, EITHER THE LEFT OR RIGHT HALF OF W, AS DETERMINED BY WHAT C/D(0) USED TO BE, I.E., ACCORDING TO THE OPPOSITE OF THE WAY C/D(0) IS NOW.	SOURCE WAS LEFT HALF C/D(0) WAS = 0 <div><div>W REGISTER</div><div>MSB LSB</div><div>↓ ↓</div><div>OMB OLB</div><div>↓ ↓</div><div>C/D(0) NOW = 1</div><div>↓ ↓</div><div>CROSS</div><div>↓ ↓</div><div>DMB DLB</div><div>↓ ↓</div><div>IDC BUS</div></div>	SOURCE WAS RIGHT HALF C/D(0) WAS = 1 <div><div>W REGISTER</div><div>MSB LSB</div><div>↓ ↓</div><div>OMB OLB</div><div>↓ ↓</div><div>C/D(0) NOW = 0</div><div>↓ ↓</div><div>CROSS</div><div>↓ ↓</div><div>DMB DLB</div><div>↓ ↓</div><div>IDC BUS</div></div>

FIG 6-2-S

SECTION 6 (CONTINUED)

picks one of bits 0 through 7. The 3 LSB Interrupt Vector Generator merely generates a 3-bit binary code corresponding to the number of the bit that was selected. The micro-instruction DMP ISC places those 3 bits onto the bottom 3 bits of the IDC Bus. At the same time, other micro-instructions and other circuitry

act to put additional information onto the IDC Bus to form the complete interrupt vector.

Figure 6-2-S illustrates, for the 16-bit version, the various uses of the W register that occur during Place and Withdraw machine-instructions. The diagram is an excellent summary of those operations, and is self-explanatory.

DETAILS OF W REGISTER CONTROL, GENERATION OF $\overline{\text{BYTE}}$ & $\overline{\text{BL}}$

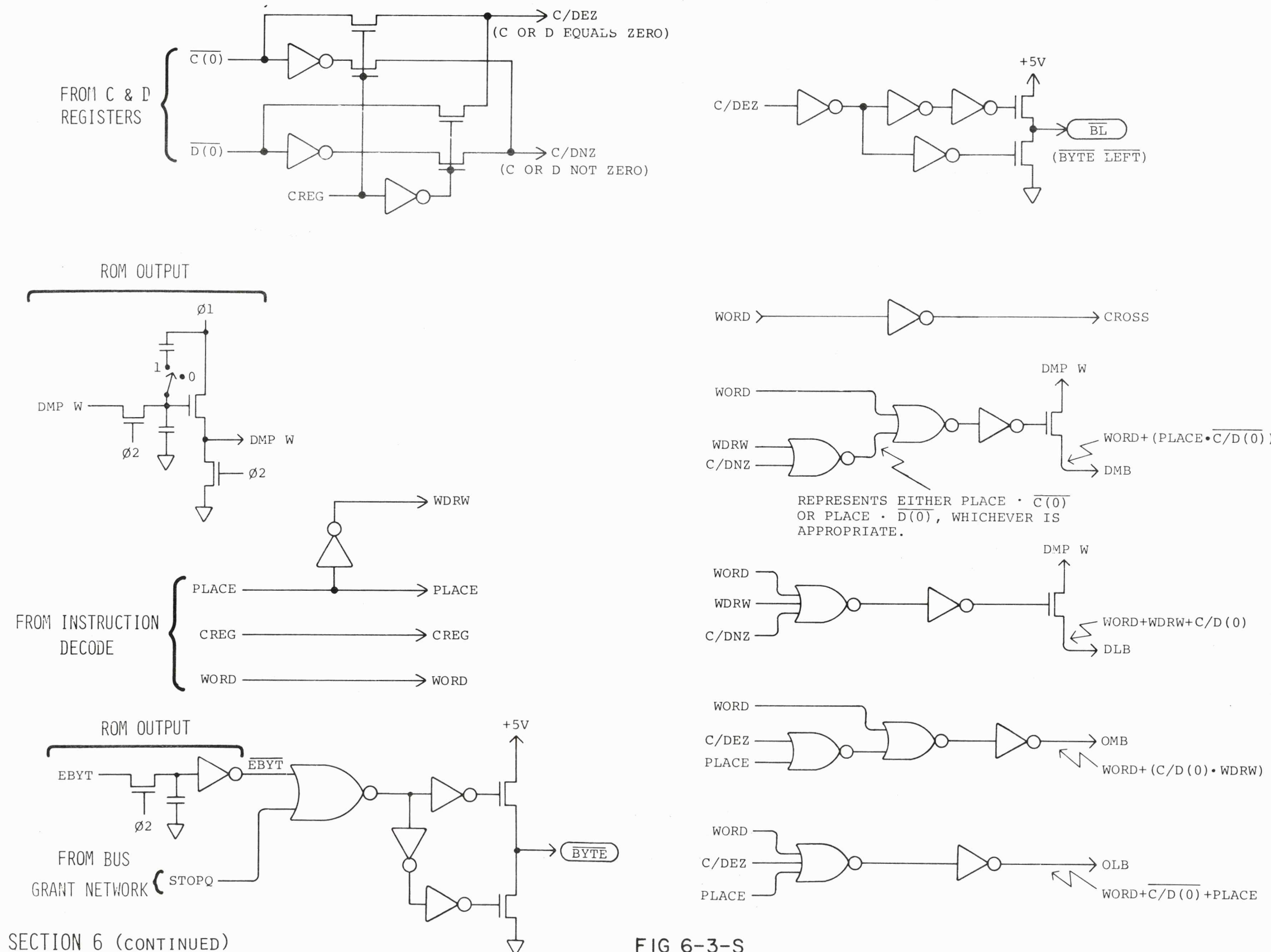


FIG 6-3-S

SECTION 6 (CONTINUED)

Figure 6-3-S shows the details of W Register Control for the 16-bit version. Observe how $\overline{\text{BL}}$ is generated. It merely represents bit 0 of the appropriate pointer register.

$\overline{\text{BL}}$ has meaning only when $\overline{\text{BYTE}}$ is false. $\overline{\text{BYTE}}$ is controlled directly by a micro-instruction decoded from the ROM. It is given only when a BYTE operation is in progress.



W REGISTER
CONTROL,
 $\overline{\text{BYTE}}$ & $\overline{\text{BL}}$



CROSSOVER &
NON-CROSSOVER
W OPERATION

TABLE OF CROSSOVER & NON-CROSSOVER W REGISTER OPERATION

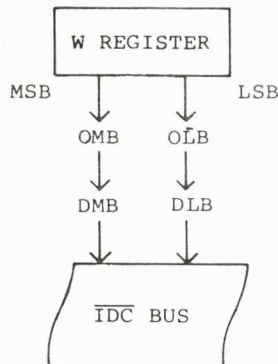
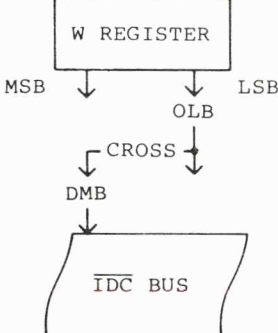
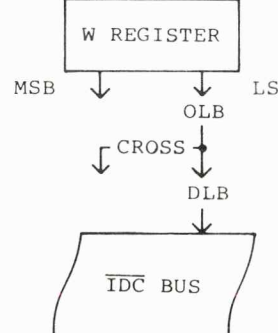
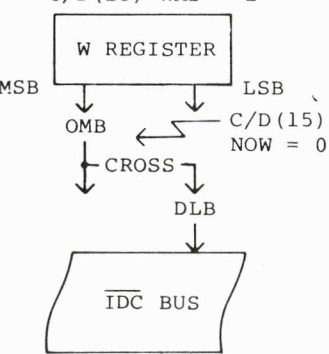
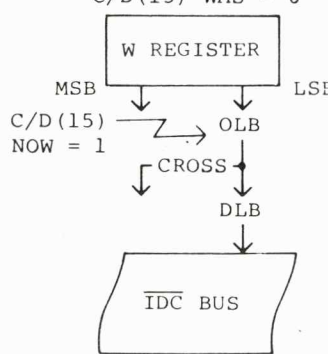
PLACE WORD		WRITING THE ENTIRE WORD OF W	
(REG. 0-7) → (C OR D INDIRECT) INCREMENT/DECREMENT C OR D BEFORE THE ACTUAL PLACE OPERATION	1. START MEMORY CYCLE TO READ SOURCE REGISTER, PUT ENTIRE DATA WORD INTO W. 2. UPDATE C OR D (BEFORE (3)). 3. START MEMORY CYCLE TO WRITE ENTIRE WORD OF W TO MEMORY AT LOCATION C OR D INDIRECT.	 <div>OMB=WORD+(WDRW·C/D(15)) OLB=WORD+PLACE+C/D(15) DMB=WORD+(PLACE·C/D(15)) DLB=WORD+WDRW+C/D(15) CROSS=WORD</div>	
WITHDRAW WORD			
(C OR D INDIRECT) → (REG. 0-7) INCREMENT/DECREMENT C OR D AFTER THE ACTUAL WITHDRAW OPERATION	1. START MEMORY CYCLE TO READ C OR D INDIRECT, PUT ENTIRE DATA WORD INTO W. 2. UPDATE C OR D (AFTER (1)). 3. START MEMORY CYCLE TO WRITE ENTIRE WORD OF W INTO DESTINATION REGISTER.		
PLACE BYTE			
(RIGHT-HAND BYTE OF REG.0-7) → (LEFT OR RIGHT HALF OF C OR D INDIRECT) C/D(15)=1⇒ LEFT HALF C/D(15)=0⇒ RIGHT HALF INCREMENT/DECREMENT C OR D BEFORE THE ACTUAL PLACE OPERATION	1. START MEMORY CYCLE TO READ REGISTER, PUT ENTIRE DATA WORD INTO W. 2. UPDATE C OR D (BEFORE (3)). 3. START A MEMORY CYCLE TO WRITE TO C OR D INDIRECT. 4. AS DATA, OUTPUT THE RIGHT HALF OF W TO EITHER THE UPPER OR LOWER HALF OF THE IDC BUS, AS DETERMINED BY C/D(15).	DESTINATION IS LEFT HALF C/D(15) IS = 1 	DESTINATION IS RIGHT HALF C/D(15) IS = 0 
WITHDRAW BYTE			
(LEFT OR RIGHT HALF OF C OR D INDIRECT) → (RIGHT HAND BYTE OF REG. 0-7) C/D(15)=1⇒ LEFT HALF C/D(15)=0⇒ RIGHT HALF INCREMENT/DECREMENT C OR D AFTER THE ACTUAL WITHDRAW OPERATION	1. START MEMORY CYCLE TO READ C OR D INDIRECT, PUT ENTIRE DATA WORD INTO W. 2. UPDATE C OR D (BEFORE (3), BUT!!). 3. C/D(15) IS NOW THE OPPOSITE OF WHAT IT WAS AT (1). 4. START A MEMORY CYCLE TO THE DESTINATION REGISTER. 5. AS DATA, OUTPUT TO THE LOWER HALF OF THE IDC BUS, EITHER THE LEFT OR RIGHT HALF OF W, AS DETERMINED BY WHAT C/D(15) USED TO BE, I.E., ACCORDING TO THE OPPOSITE OF THE WAY C/D(15) IS NOW.	SOURCE WAS LEFT HALF C/D(15) WAS = 1 C/D(15) NOW = 0 	SOURCE WAS RIGHT HALF C/D(15) WAS = 0 C/D(15) NOW = 1 

FIG 6-2-F

SECTION 6 (CONTINUED)

Figure 6-2-F illustrates the various means for dumping the W register during 15-bit operation. The end result is the same as in the 16-bit case, but the method varies slightly.

DETAILS OF W REGISTER CONTROL, AND GENERATION OF $\overline{\text{BYTE}}$

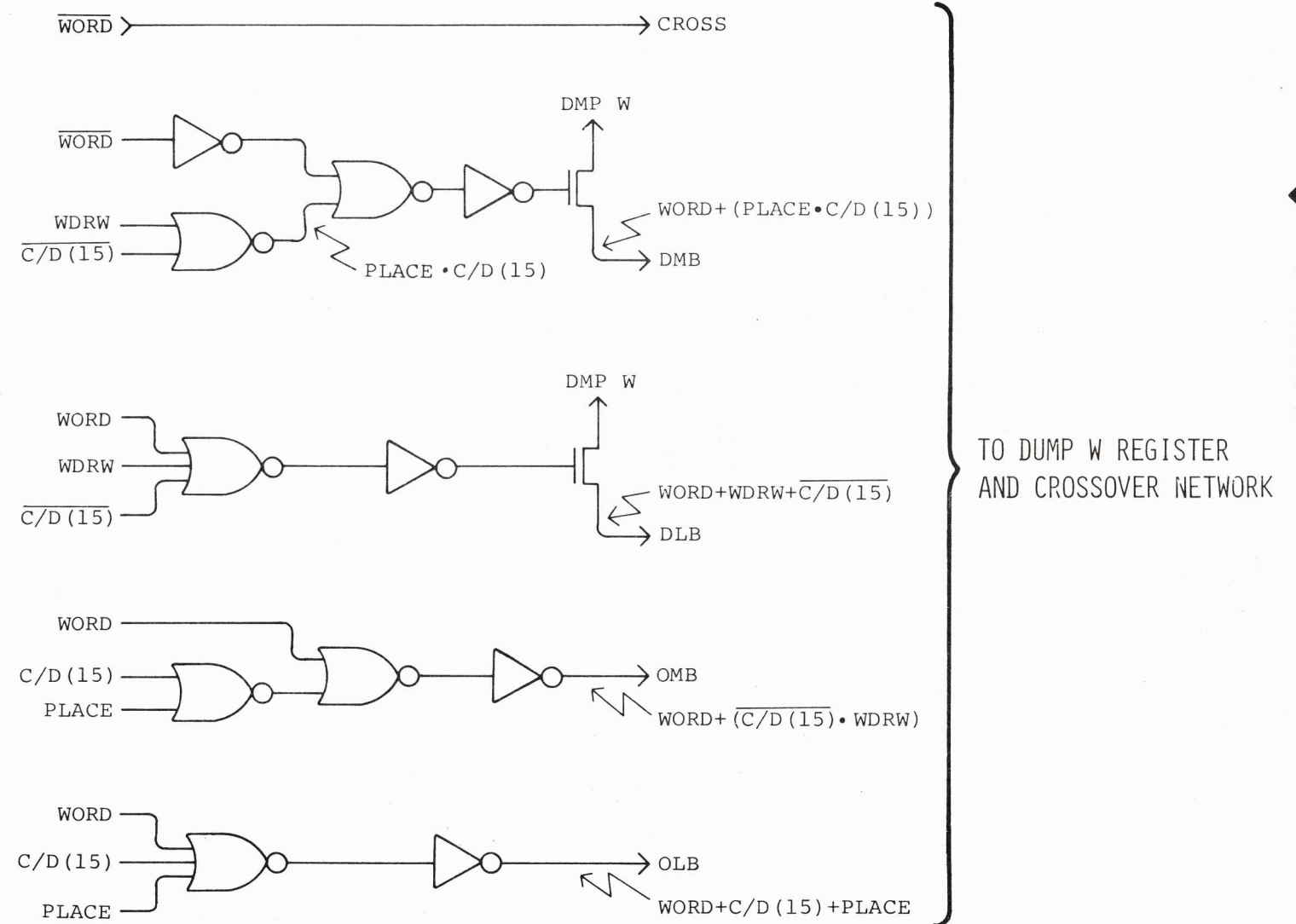
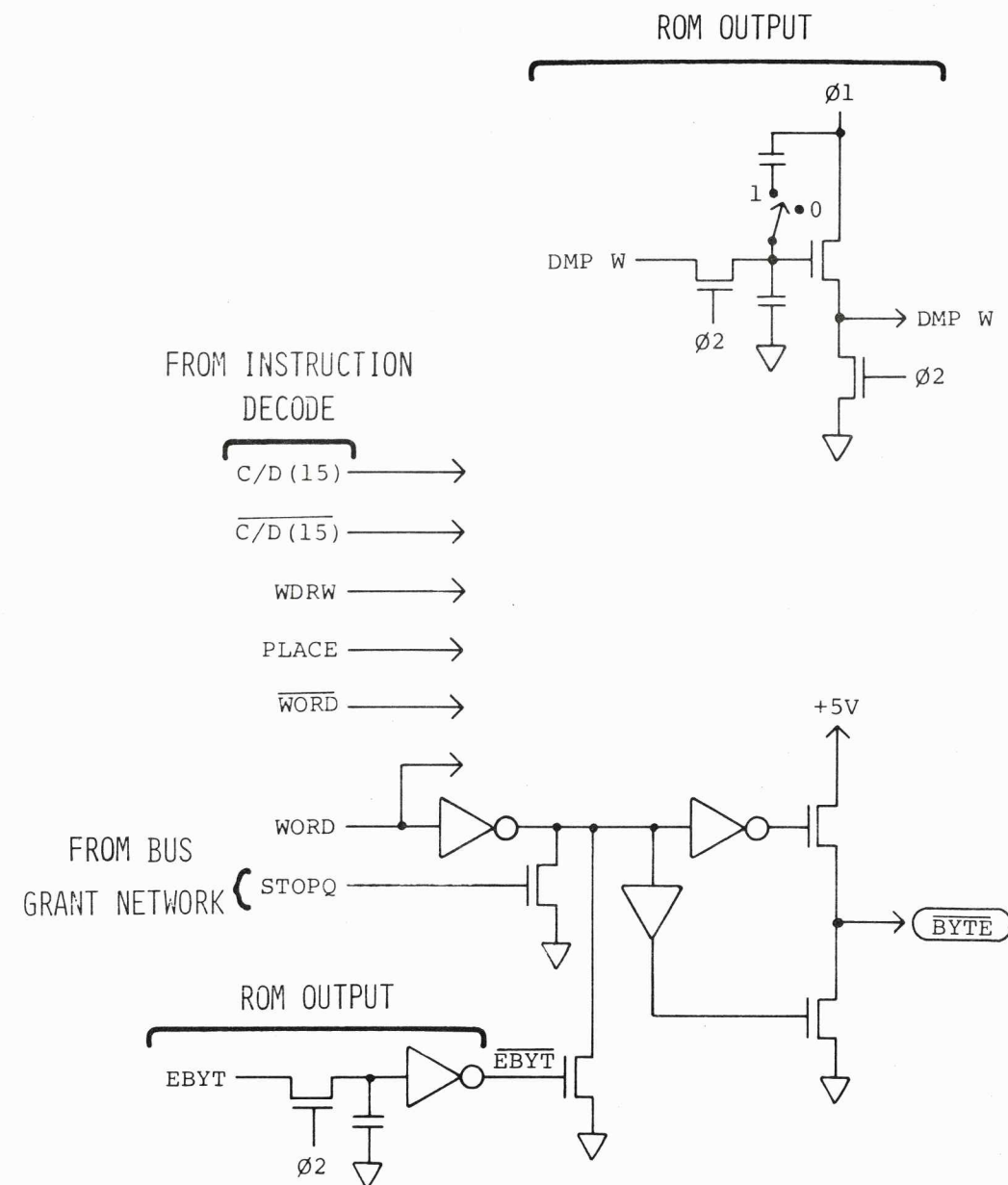
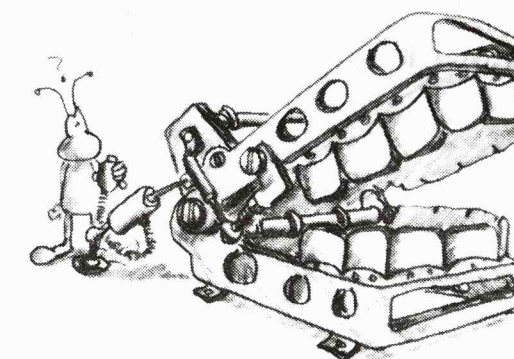


FIG 6-3-F

SECTION 6 (CONTINUED)

Figure 6-3-F illustrates the details of W Register Control for the 15-bit version. One of the main differences between the two versions is that in the 15-bit version it is bit 15 of the associated pointer register that controls the "cross" function. In the 16-bit version it is bit 0.

Also, note the absence of the signal $\overline{\text{BL}}$ and minor differences in the $\overline{\text{BYTE}}$ circuitry.



CONTROLLING
THE
BYTE LINE

W REGISTER
CONTROL,
BYTE

CROSSOVER &
NON-CROSSOVER
W OPERATION



FIG 6-4



Figure 6-4 illustrates the details of the W register proper, and of the SET W circuitry.

Figure 6-5 illustrates in detail the exact means of implementing OMB, OLB, CROSS, DMB and DLB.

DETAILS OF THE INTERRUPT VECTOR NETWORK

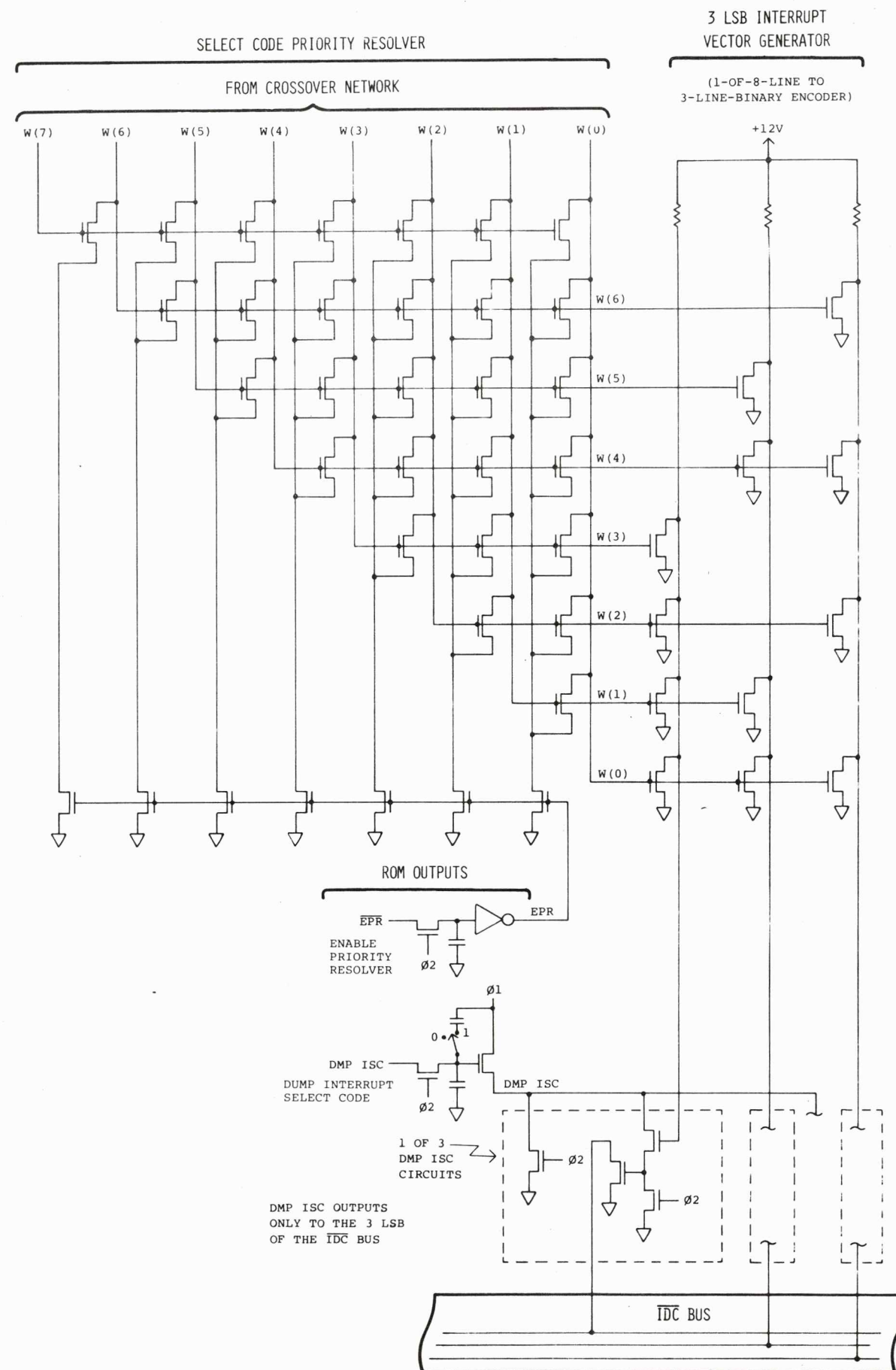


FIG 6-6

SECTION 6 (CONTINUED)

Figure 6-6 shows the details of the Select Code Priority Resolver and of the 3 LSB Interrupt Vector Generator. Note that the micro-instruction EPR (Enable Priority Resolver) must be issued to activate this circuitry. Also note that DMP ISC (Dump Interrupt Select Code) is necessary to place the encoded partial interrupt vector onto the IDC Bus.

INTERRUPT
VECTOR NETWORK

DUMP W &
CROSSOVER CIRCUITS

W REGISTER
AND SET W

OVERVIEW OF THE IV REGISTER

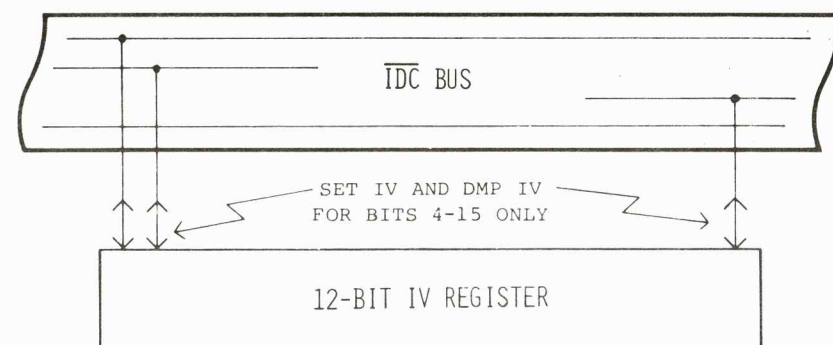


FIG 7-1

DETAILS OF THE IV REGISTER

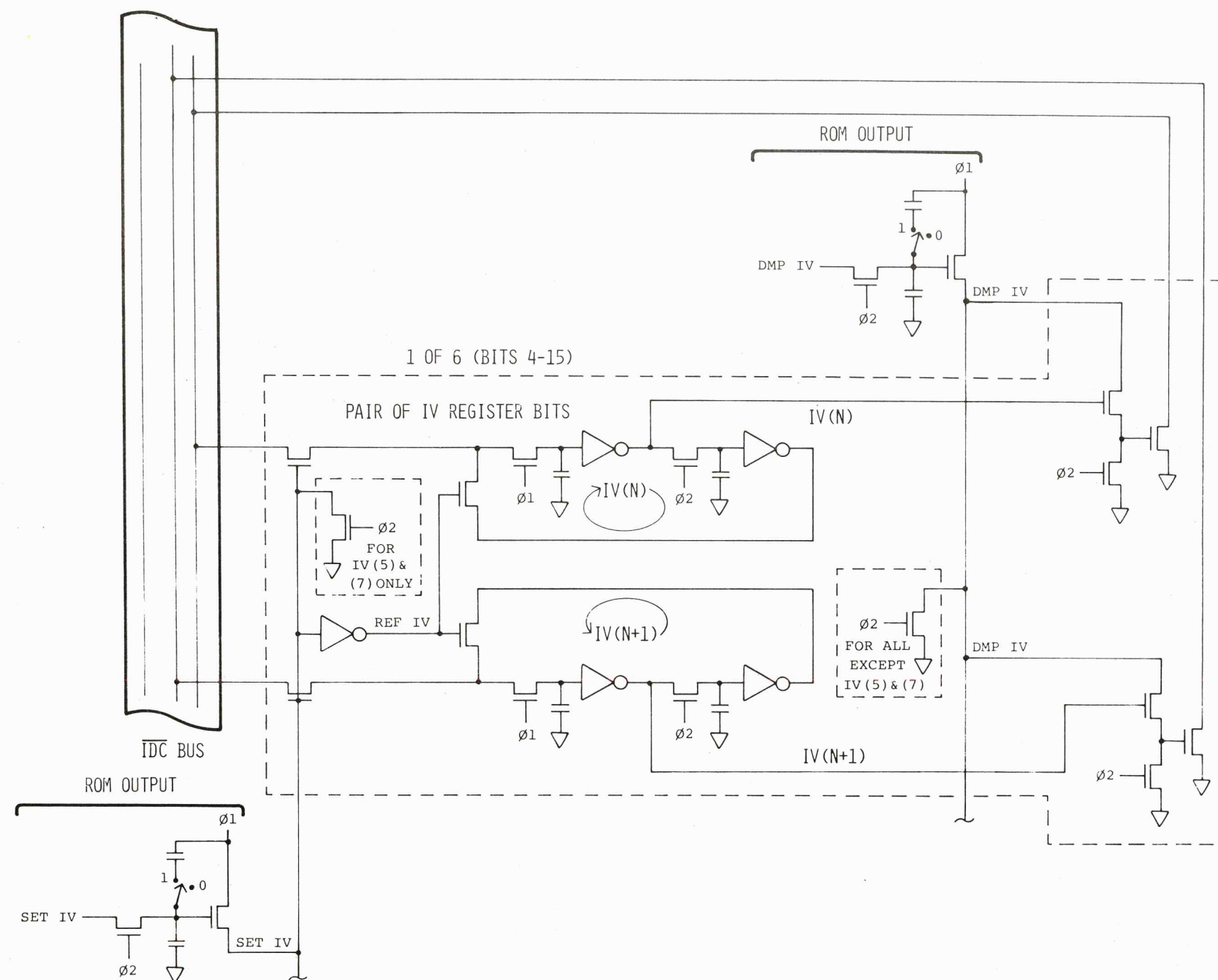


FIG 7-2

SECTION 7

Figure 7-1 is an overview of the IV register. The IV register has only 12 bits, numbered 4 through 15. The IV register is intended to contain only the upper 12 bits of the address for the interrupt vector. The bottom 4 bits are supplied by another mechanism, and are determined according to the conditions of the interrupt. The micro-instructions SET IV and DMP IV are quite ordinary.

Figure 7-2 shows the details of the individual cells of the IV register. There is nothing particularly remarkable about this circuitry.

OVERVIEW OF THE PERIPHERAL ADDRESS STACK & PERIPHERAL ADDRESS STACK CONTROL

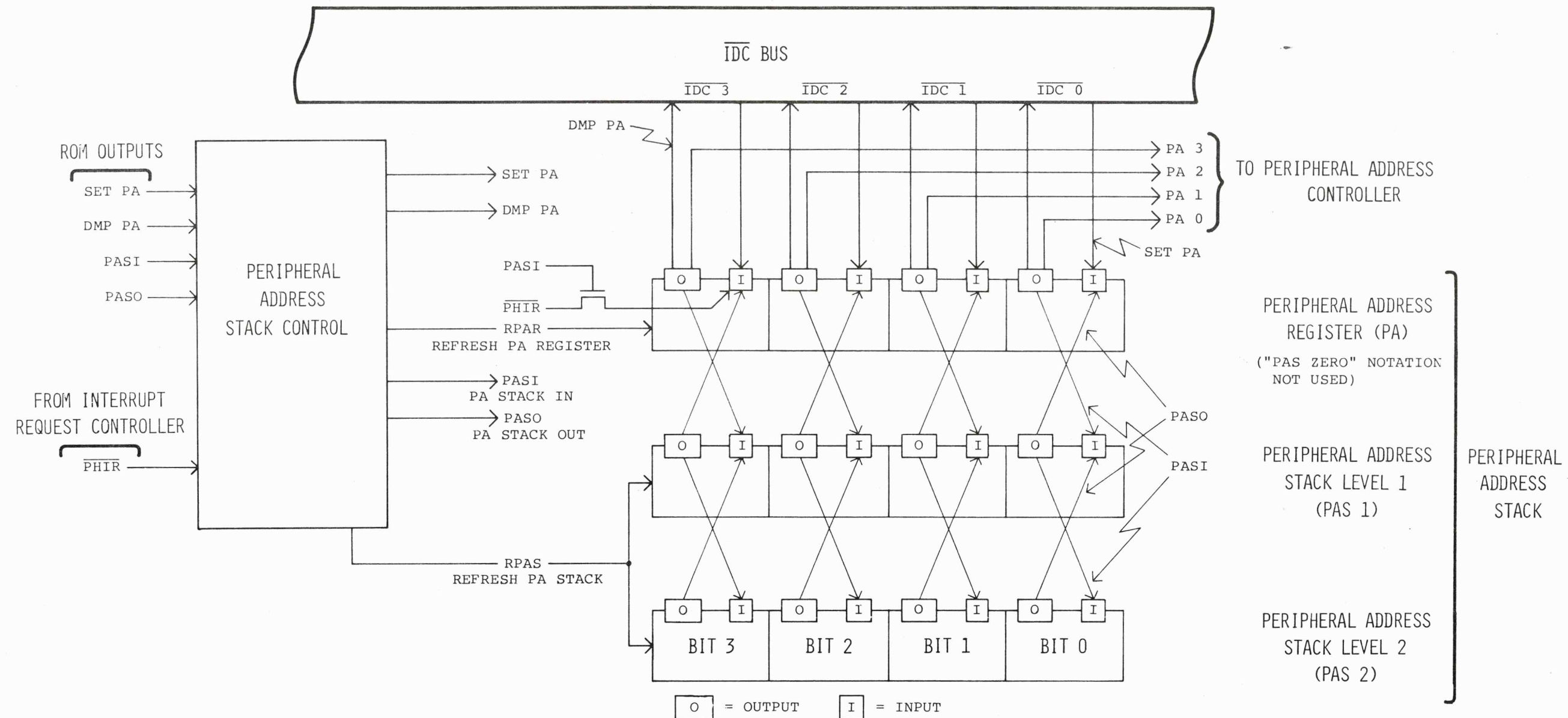


FIG 8-1

SECTION 8

Figure 8-1 is an overview of the Peripheral Address Stack mechanism. The stack is 4 bits wide and 3 levels deep. The top level of the stack functions as the PA register. The stack contains select codes. When an interrupt occurs the previous select code-in-use is stacked. The new (interrupting) select code is placed in PA. As interrupts are satisfied the stack is popped and the select code-in-use becomes what it was before the interrupt.

Thus, the peripheral address

stack is used in two ways. First, its contents are used to determine the address presented by the Peripheral Address Bus, except when DMA operations are in progress. The firmware programmer controls the *initial* value of PA when he, for example, indulges in so-called simple I/O. Second, PA is set as a result of an interrupt poll. The previous value will be pushed on the stack. A SET PA will set the lower 3 bits of PA to the value generated by the Select Code Priority Resolver and the 3 LSB Interrupt Vector Generator.

The 4th bit of PA will be set according to the level of interrupt being allowed. This is arranged by having PASI load the value of PHIR into the 4th bit of PA. PHIR comes from the Interrupt Controller and represents the level of interrupt in being requested. (It is also the same thing as the 4th bit of the peripheral address or select code.)

The PA register itself can be set and dumped in the ordinary manner. A new entry is pushed onto the stack as follows. To begin with, the new entry itself is put into PA

with a SET PA. At the same time the micro-instruction PASI causes the previous entry in PA to move to the next lower level of the stack (PAS 1). The same PASI shifts the contents of PAS 1 into PAS 2. The micro-instruction PASO is used to POP the stack. It shifts everything up one level. Generally speaking, a DMP PA does not accompany a PASO.

The new contents of the PA register are available by direct connection to the Peripheral Address Controller.

PA STACK &
PA STACK CONTROL
OVERVIEW

IV REGISTER
OVERVIEW

DETAILS OF PERIPHERAL ADDRESS STACK CONTROL

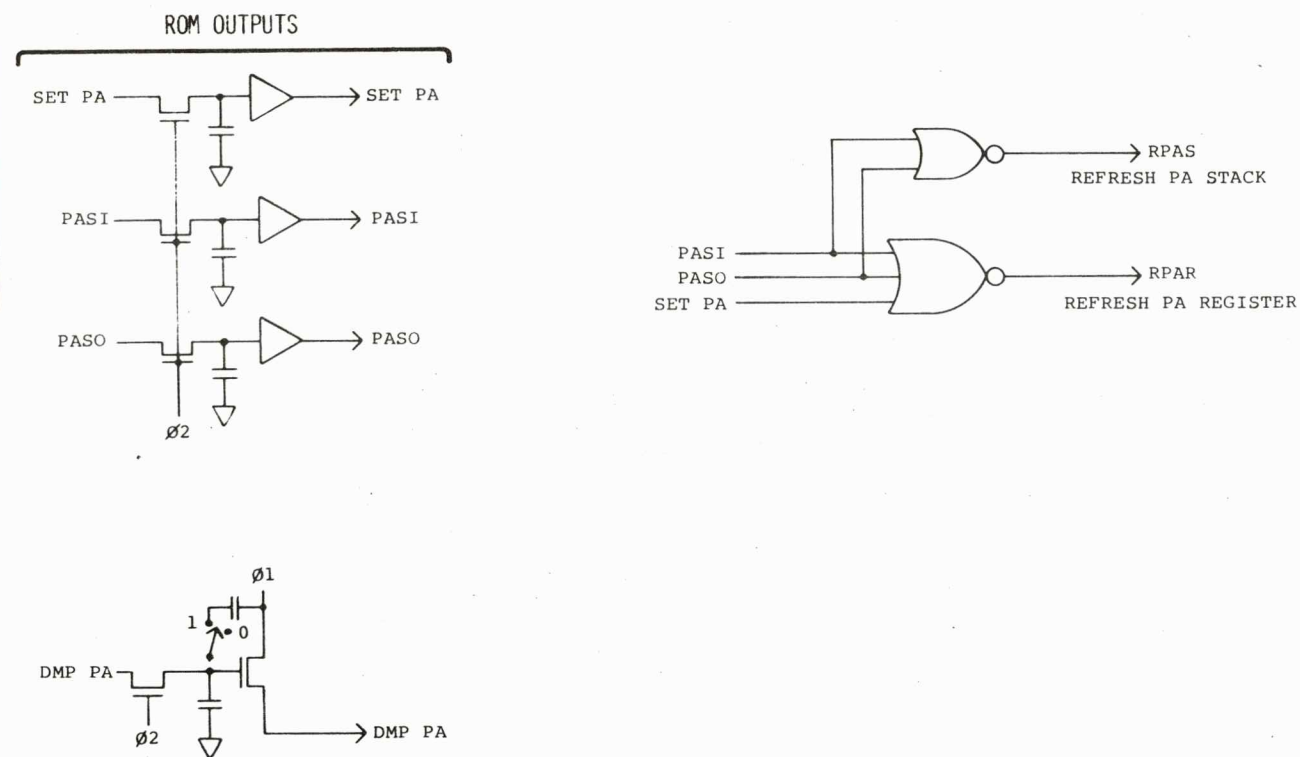


FIG 8-2

SECTION 8 (CONTINUED)

Figure 8-2 shows the details of the Peripheral Address Stack Control circuitry. Observe that whenever the stack is being neither pushed nor popped that RPAS is generated. That refreshes the bottom two levels of the stack. Likewise, when the stack is being neither refreshed nor popped, and PA is not being set, RPAPR is generated. That refreshes the PA register.

Figure 8-3 shows the details of the individual cells of the Peripheral Address Stack.

DETAILS OF THE PERIPHERAL ADDRESS STACK

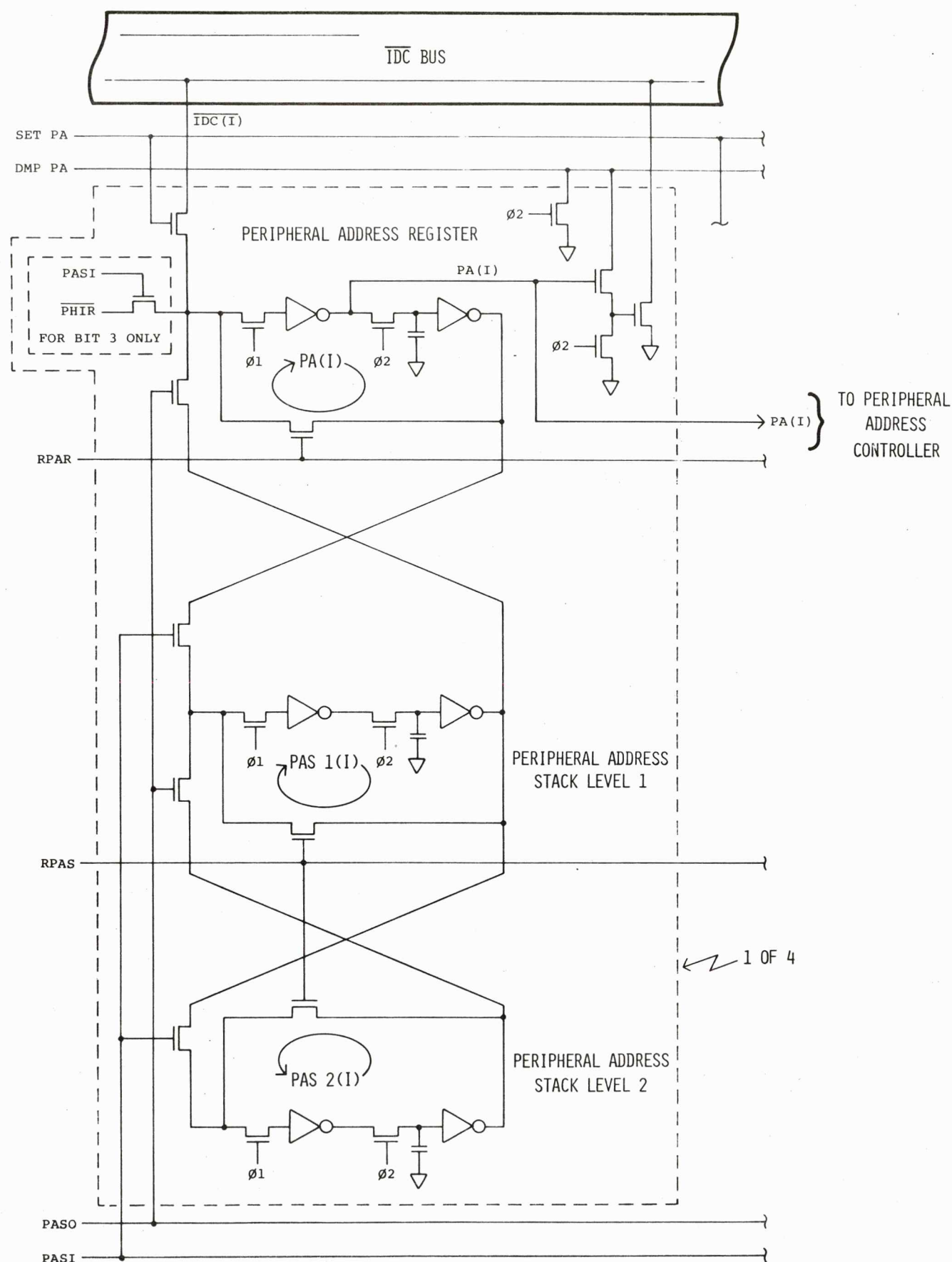


FIG 8-3

OVERVIEW OF THE DMAC REGISTER

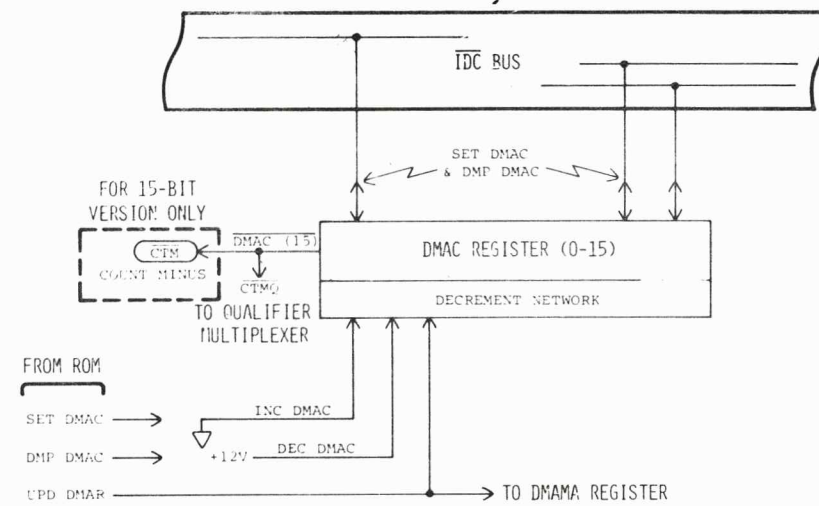


FIG 9-1

SECTION 9

Figure 9-1 is an overview of the DMAC register. In both versions DMAC is a 16-bit register whose purpose is to assist in controlling the number of data transfers involved in any particular DMA operation. The usual mode of operation is for DMAC to be set in advance to the number of desired transfers minus one. As each transfer is performed DMAC is decremented one count. When the count in DMAC is negative a qualifier (CTMQ) is generated to tell the Bus Controller that the DMA operation is completed. In addition, under these circumstances the 15-bit version also generates CTM (Count Minus). This is a signal whose original intent was to inform the peripheral that the operation is completed.

However, there were troubles with CTM, and its use has been discontinued.

DMAC can be set and dumped in the ordinary manner. The micro-instruction UPD DMAR is used to cause DMAC to decrement one count. Notice that UPD DMAR is sent to another register besides DMAC. That other register is DMAMA, and it uses this command to increment in the same manner that DMAC uses it to decrement.

DMAC BIT 0 AND DMAC CONTROL SIGNALS

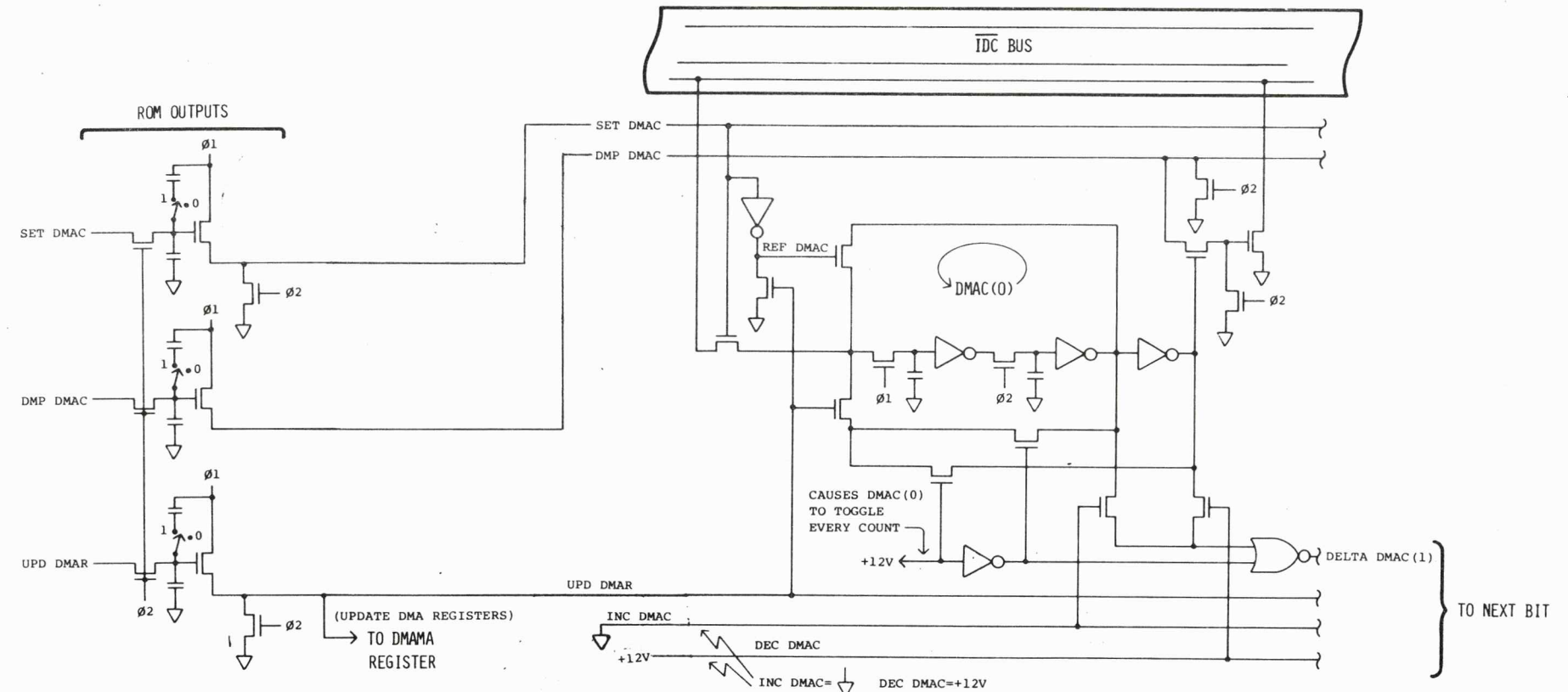


FIG 9-2

Figure 9-2 shows the details of the least significant bit of DMAC, and of the generation of the micro-instructions and control signals for DMAC. Observe that bit 0 has been connected so that it toggles every count. Also observe that DMAC contains hardware to increment as well as to decrement. The increment line, however, is permanently grounded, just as the decrement line is permanently made high. There is no special reason for this, other than it was easy to lay it out in this way. The circuitry for the C and D registers was taken as a model for DMAC (and also DMMA), and it was easier to leave the unnecessary increment circuitry in place than it was to take it out.

DMAC BIT 0
& DMAC
CONTROL SIGNALS

DMAC
OVERVIEW

PA STACK

PA STACK
CONTROL

DETAILS OF THE DMAC REGISTER, BITS 1-14

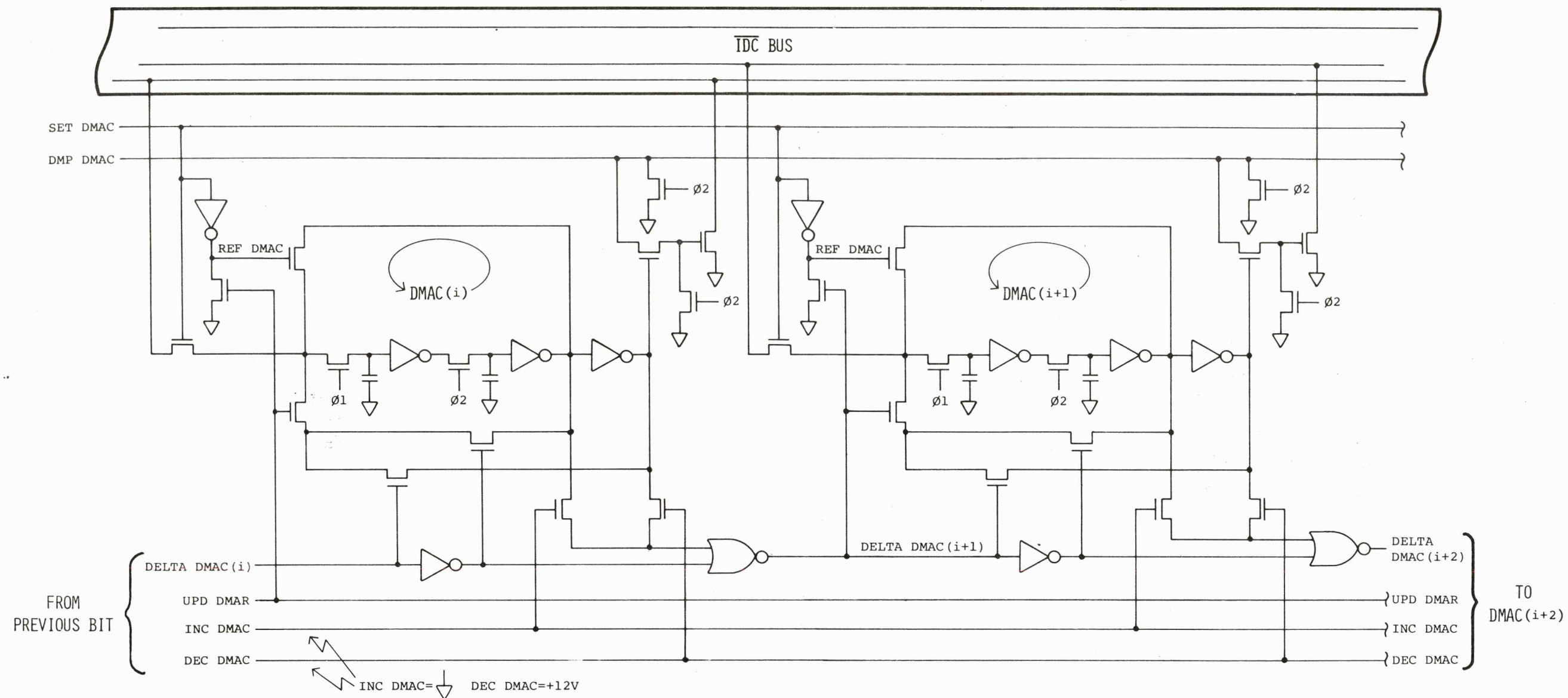


FIG 9-3

SECTION 9 (CONTINUED)

Figure 9-3 shows the details of bits 1 through 14 of DMAC. There is nothing particularly remarkable about this circuitry.

DETAILS OF DMAC REGISTER BIT 15 AND CTM

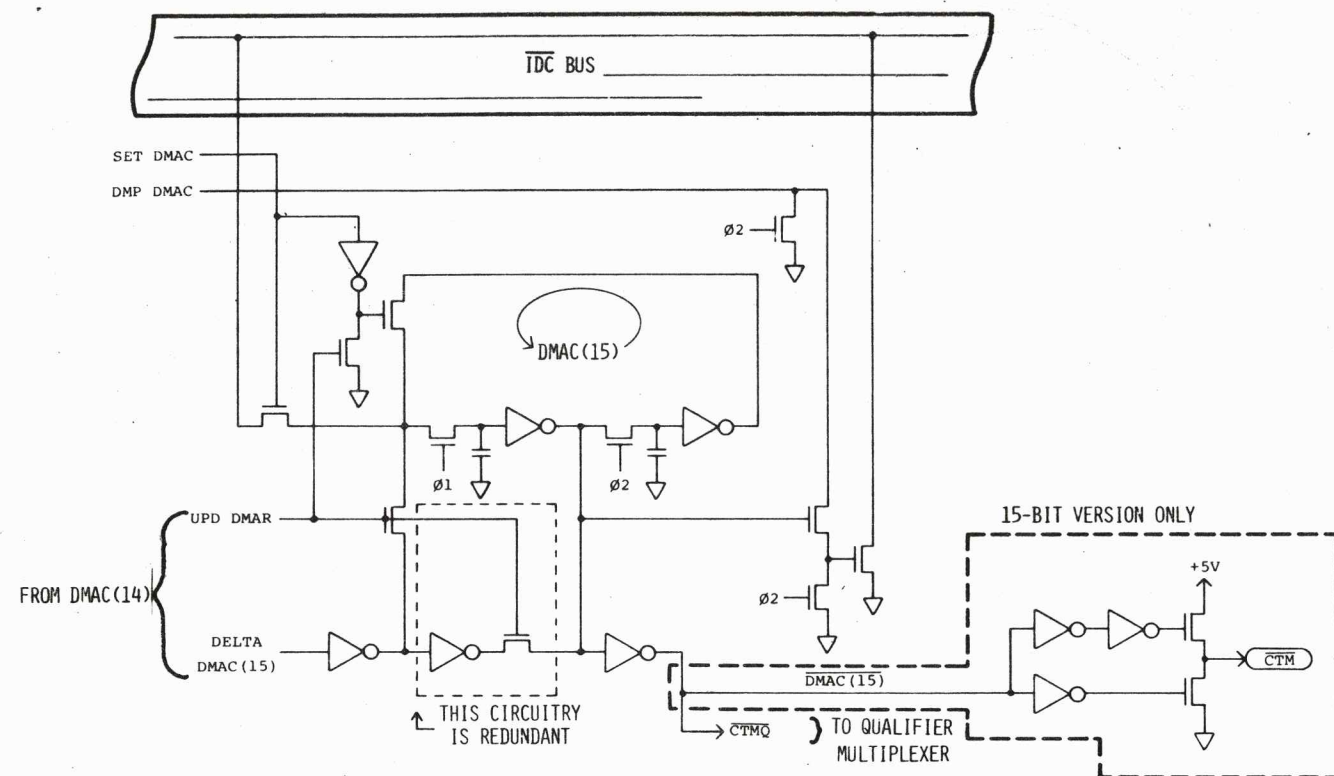


FIG 9-4

SECTION 9 (CONTINUED)

Figure 9-4 illustrates bit 15 of DMAC. Observe the manner in which bit 15 of DMAC is used to generate CTM in the 15-bit version. The idea is that when DMAC has been decremented through 0 to a negative value bit 15 will be set, thus generating CTM.

As for the redundant circuitry shown, the engineer involved expressed amazement that it was there in the first place. No one has been able to suggest a purpose for that circuitry. Someone proposed that it allows faster generation of CTMQ by eliminating the delay through phase one of the first half of DMAC(15). But this idea has to be discarded as soon as one realizes that UPD DMAR is not true except during phase one either. A can of Coor's to the first person who figures out what this circuitry is for, if indeed it is for anything.

OVERVIEW OF THE DMAMA AND DMAD REGISTERS

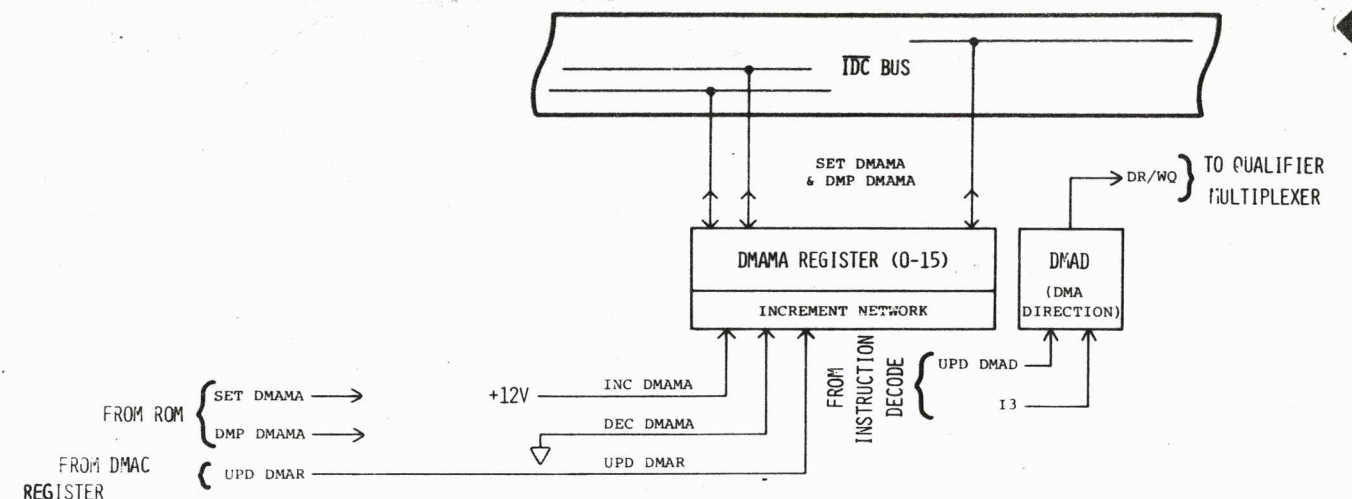


FIG 10-1-S

SECTION 10-S

Figure 10-1-S is an overview of the DMAMA and DMAD registers, as employed in the 16-bit version. DMAMA is a 16-bit register used to hold the memory address associated with ongoing DMA operations. That address changes each DMA memory cycle. Accordingly, DMAMA is equipped with an Increment Network. The Increment Network is in reality an increment/decrement network, but the increment control line has been made permanently high and the decrement line has been permanently grounded. DMAMA will increment once upon receipt of each UPD DMAR.

The micro-instructions SET DMAMA and DMP DMAMA are quite straightforward.

The DMAD register is a 1-bit register used to indicate the direction of DMA transfers. DMAD is changed only in response to the SDO and SDI machine-instructions. Such

machine-instructions generate the micro-instructions UPD DMAD. Its effect is to cause DMAD to assume the value of bit 3 of the I register. As you might expect, bit 3 is exactly the difference between the SDO and SDI machine-instructions. DMAD represents its information to the ROM in the form of the qualifier DR/WQ, which stands for DMA Read/Write Qualifier.

DETAILS OF DMAMA REGISTER BIT 0 & DMAMA CONTROL SIGNALS

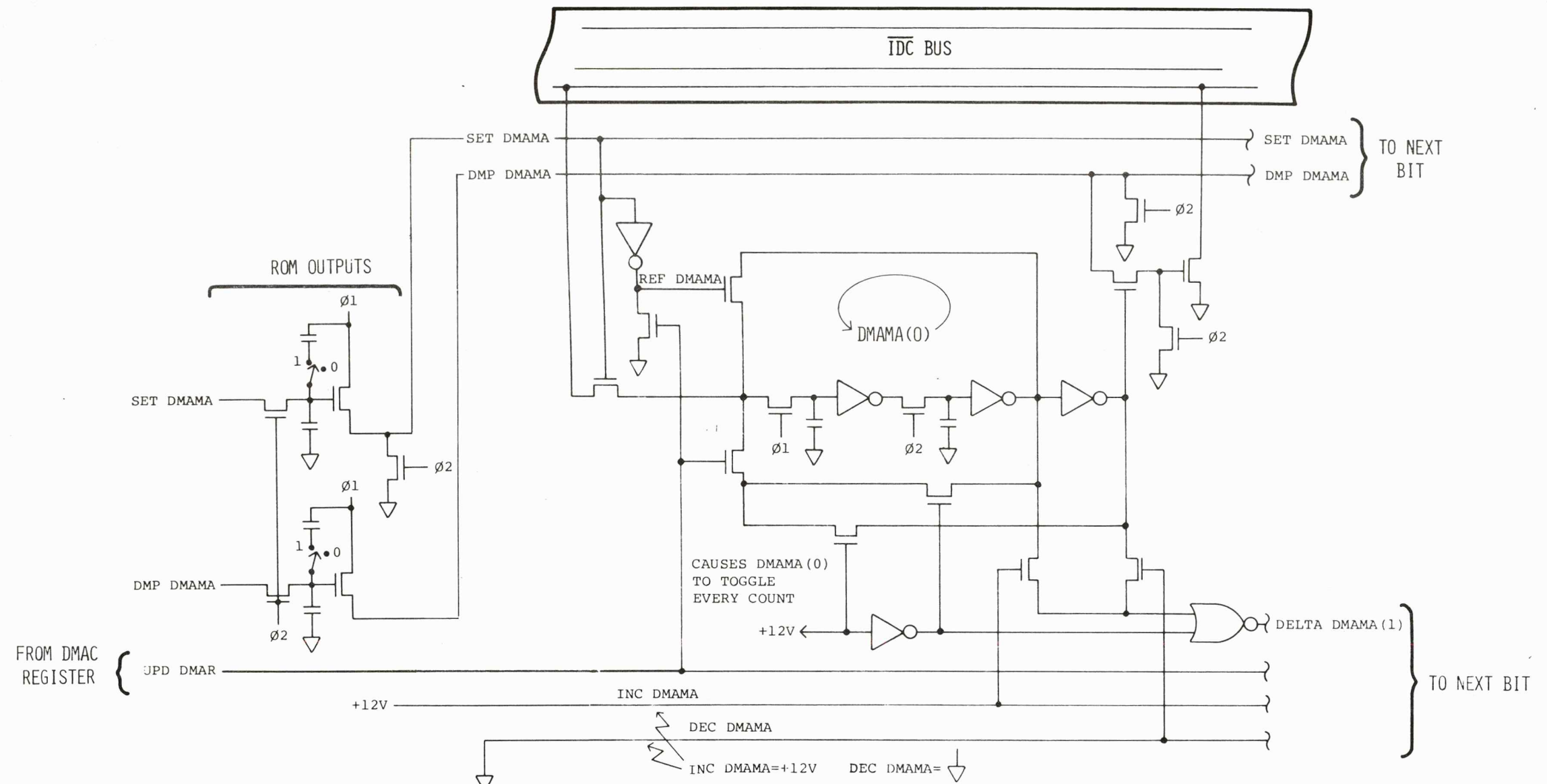


FIG 10-2-S

SECTION 10-S (CONTINUED)

Figure 10-2-S illustrates the details of the micro-instructions associated with DMAMA and of bit 0 of that register. Observe that hardware for decrementing DMAMA is in place, but that the decrement line has been permanently grounded. This was done for ease in layout of the register.



DMAMA

DETAILS OF DMAMA REGISTER BITS 1-15

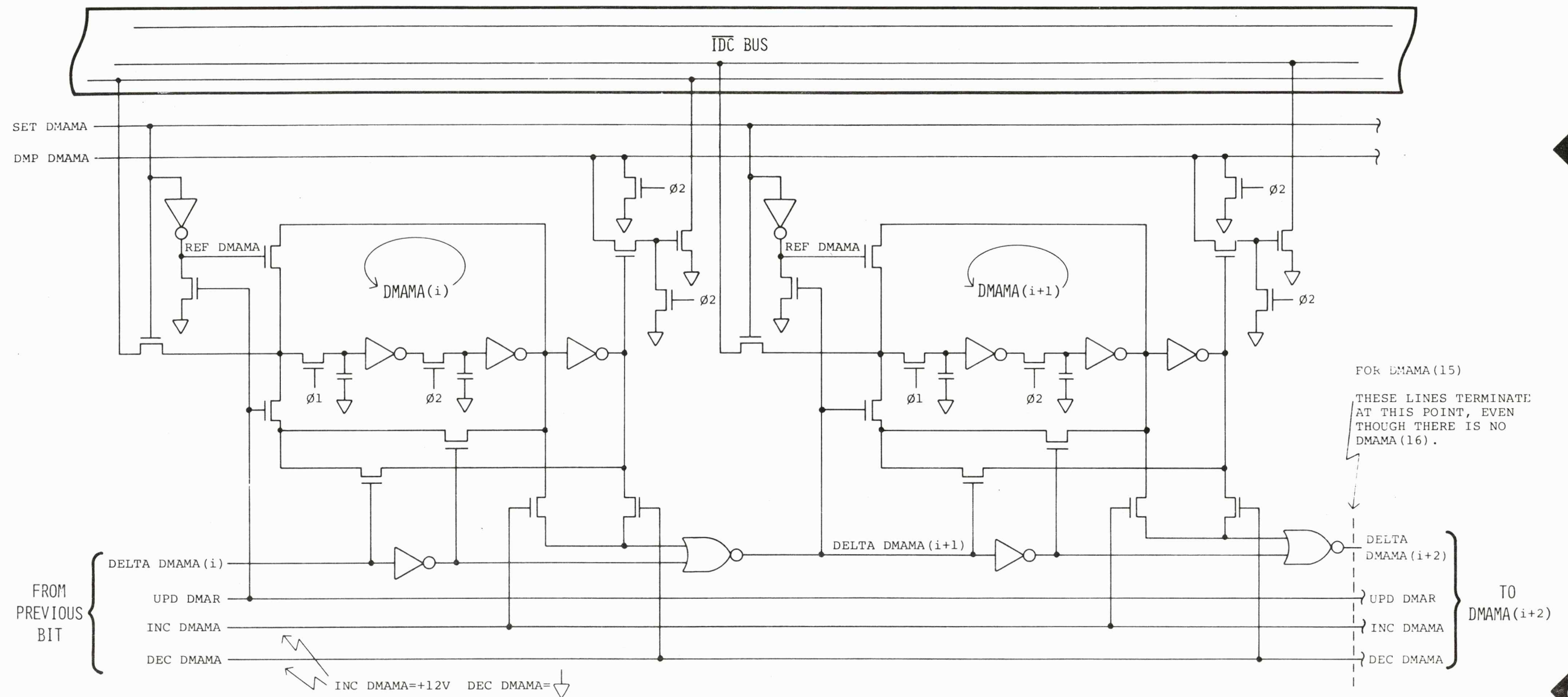


FIG 10-3-S

SECTION 10-S (CONTINUED)

Figure 10-3-S illustrates bits 1 through 15 of the DMAMA register.

DETAILS OF DMAD

16-BIT VERSION ONLY

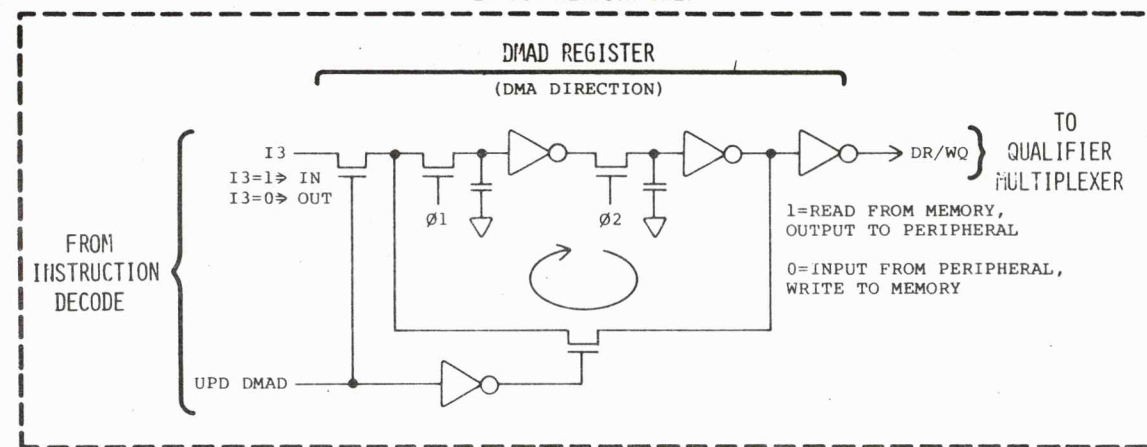


FIG 10-4-S

SECTION 10-S (CONTINUED)

Figure 10-4-S illustrates the details of the DMAD register. Observe that there is absolutely no way to interrogate this register from outside the chip.

OVERVIEW OF THE DMAMA REGISTER

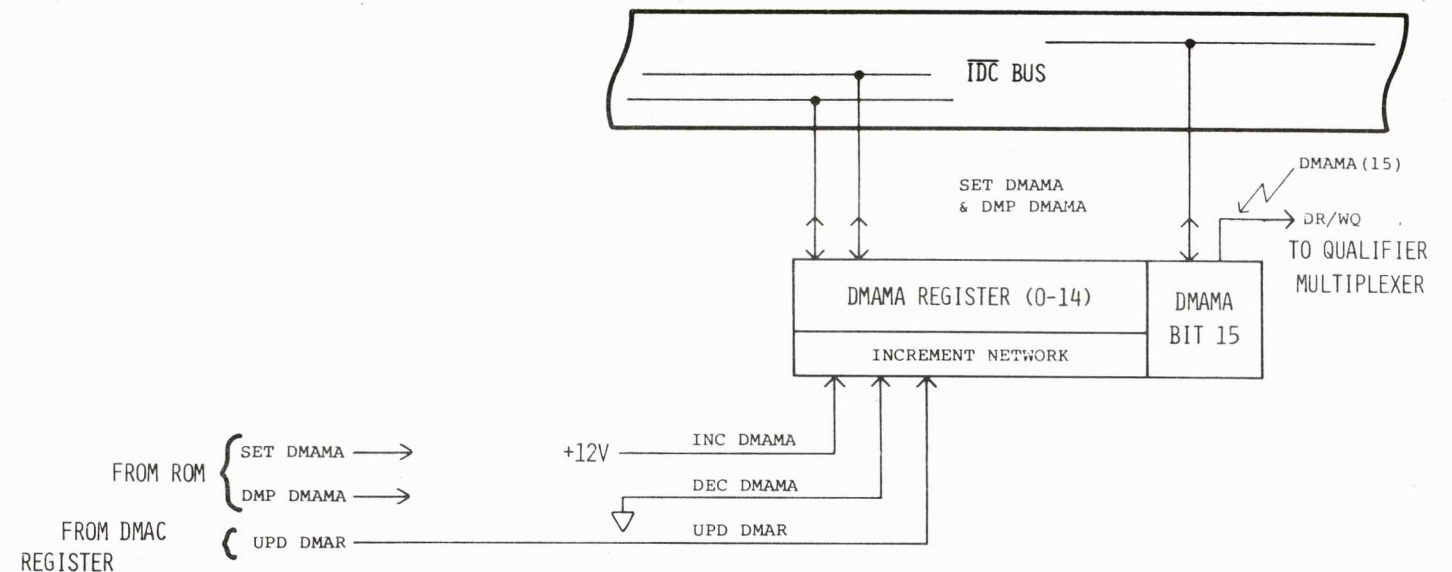


FIG 10-1-F

SECTION 10-F

Figure 10-1-F is an overview of the DMAMA register in the 15-bit version. The purpose of the DMAMA register is to generate the addresses used in DMA memory operations and to control their direction. To this end, the lower 15 bits of the DMAMA register are connected to an Increment Network. This network is actually an increment/decrement network whose decrement line has been permanently grounded and whose increment line has been permanently made high. Since only 15 bits of address are ever required, the 16th bit of DMAMA is used to represent the direction of the transfer. Accordingly, it is the source of the qualifier DR/WQ.

The Increment Network will increment the value of the DMAMA register once each time the micro-instruction UPD DMAR is received. The micro-instructions SET DMAMA and DMP DMAMA are quite straightforward. They are full 16-bit instructions. Therefore, in the 15-bit version it is possible to interrogate the direction bit of the DMA control word.

DETAILS OF DMAMA REGISTER BITS 0 AND 15

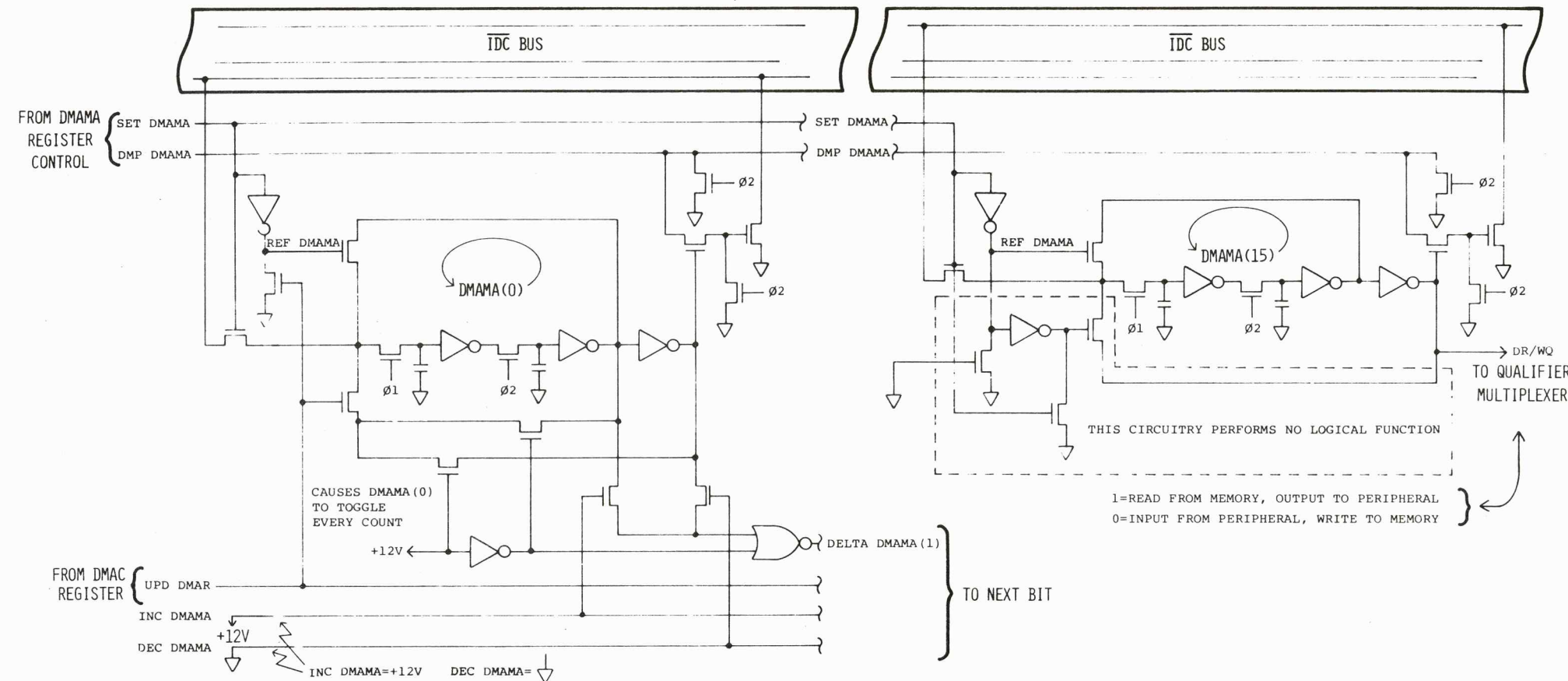


FIG 10-2-F

DETAILS OF DMAMA REGISTER BITS 1-14

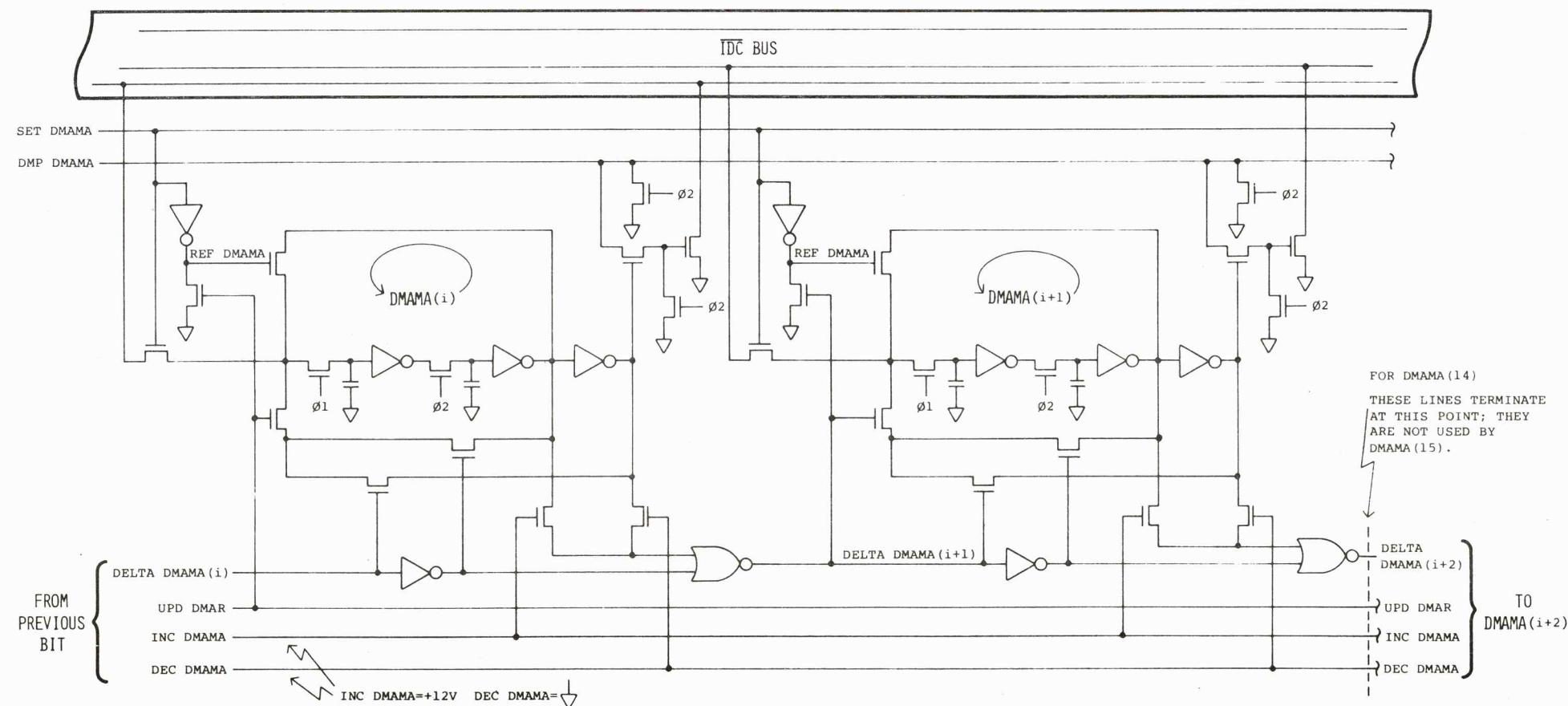


FIG 10-3-F

SECTION 10-F (CONTINUED)

Figure 10-2-F illustrates the details of bit 0 and bit 15 of DMAMA. Bit 15 has associated with it some redundant circuitry that has no logical function. Nobody has ever claimed that he knew why that part is the way it is, but a safe guess is that it has something to do with short-cuts during layout.

Figure 10-3-F illustrates the details of the micro-instructions associated with DMAMA, and the details of bits 1 through 14 of that same register. Observe that the increment mechanism is actually an increment/decrement mechanism whose decrement line has been permanently grounded and whose increment line has been permanently made high.

DMAMA REGISTER
BITS 1-14

DMAMA REGISTER
BITS 0 AND 15

DMAMA REGISTER
OVERVIEW

DMAD

OVERVIEW OF THE DMAPA REGISTER

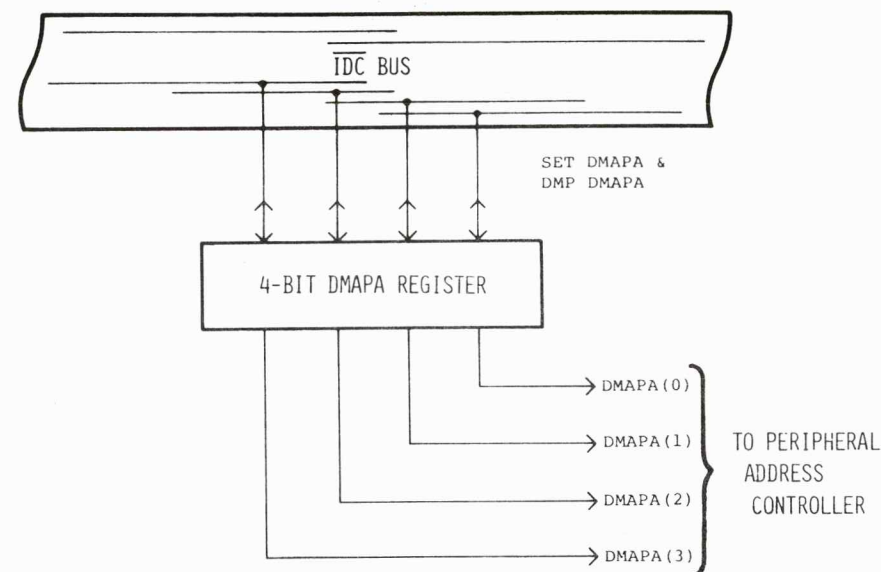


FIG II-1

DETAILS OF THE DMAPA REGISTER

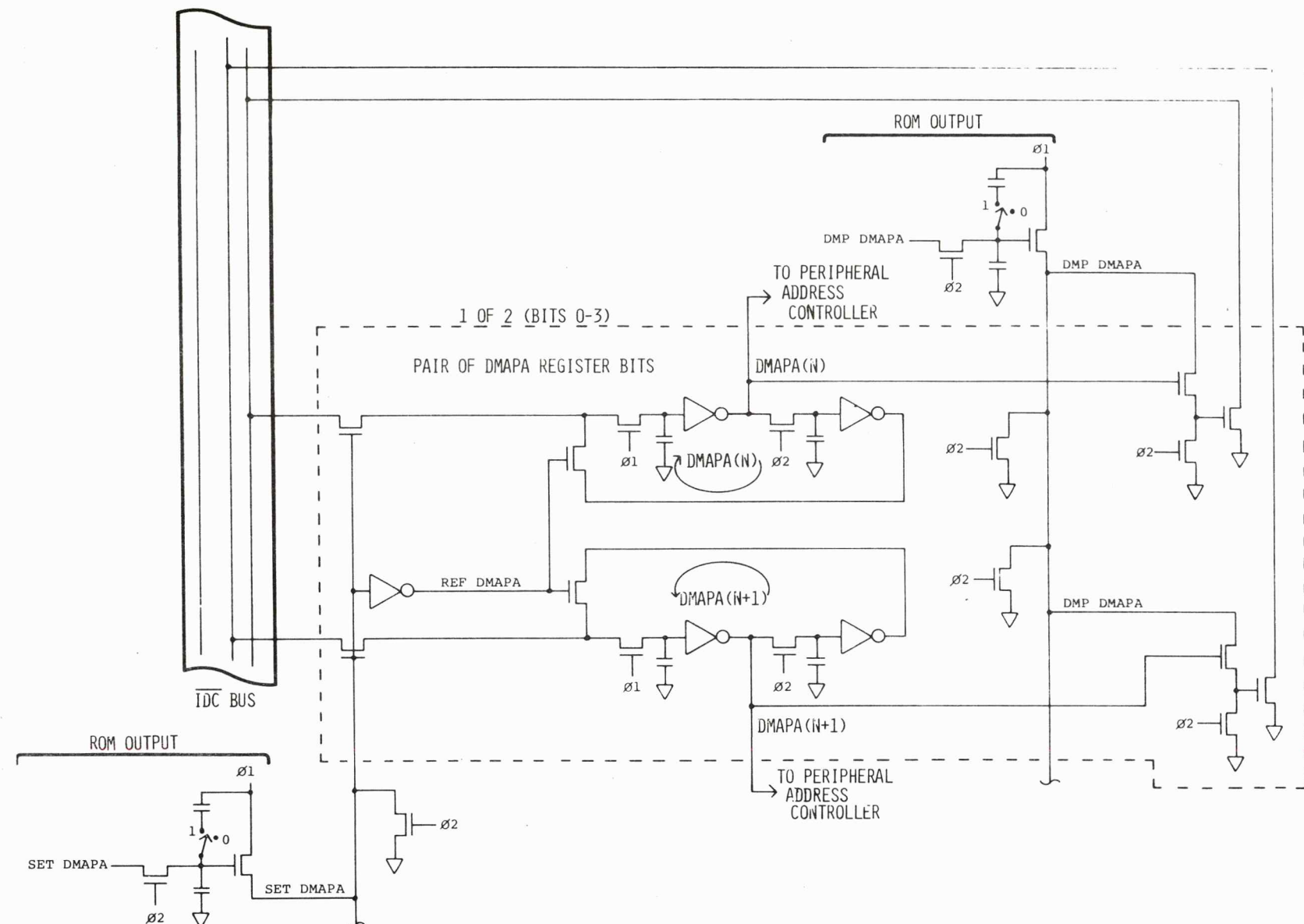


FIG II-2

SECTION 11

Figure 11-1 is an overview of the DMAPA register. The purpose of the DMAPA register is to contain the select code for peripherals involved in DMA operation. So that it may be initially set and later interrogated, both SET DMAPA and DMP DMAPA are implemented. They are quite straightforward. To generate a select code based on the contents of the register, its contents go via a direct connection to the Peripheral Address Controller.

Figure 11-2 shows the details of the DMAPA register. Aside from its direct connection to the Peripheral Address Controller, this register is rather ordinary.

OVERVIEW OF THE PERIPHERAL ADDRESS CONTROLLER

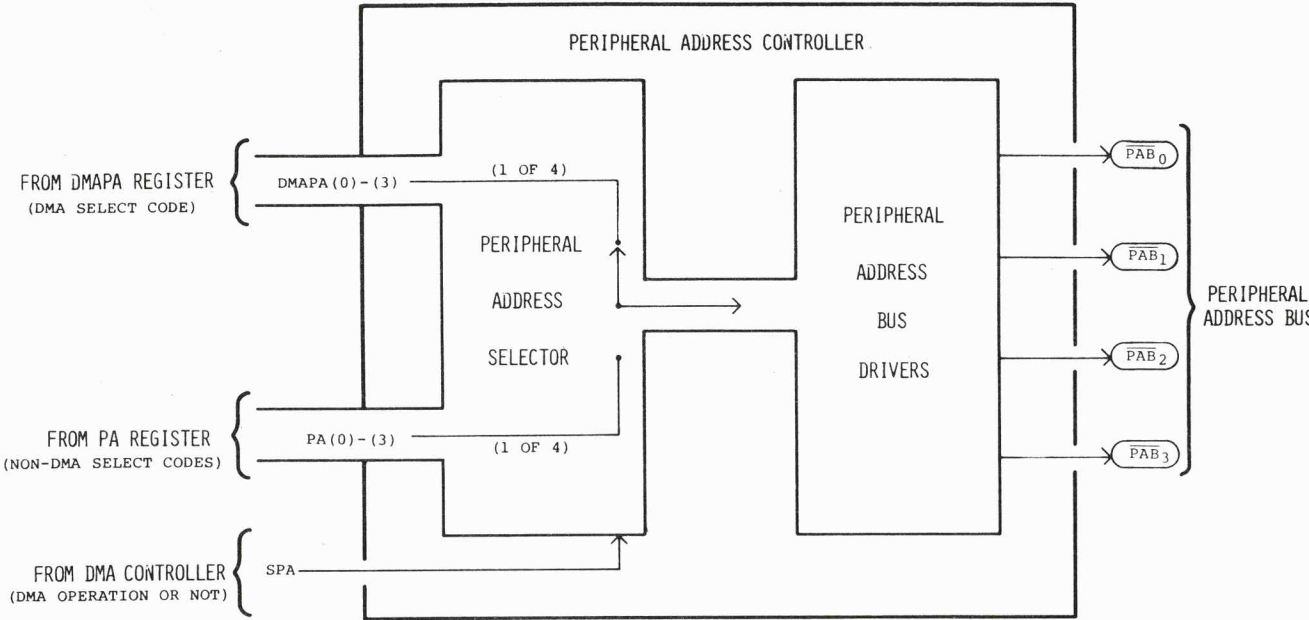


FIG 12-1

SECTION 12

Figure 12-1 is an overview of the Peripheral Address Controller. The purpose of the Peripheral Address Controller is to establish the select code-in-use and drive the Peripheral Address Bus accordingly. The Peripheral Address Controller chooses from two sources of select codes. These are the PA register and its associated stack, and the DMAPA register. The choice is made on the basis of whether or not a DMA operation is in progress. A Peripheral Address Selector makes the choice. The Peripheral Address Bus Drivers then apply the actual select code to the Peripheral Address Bus.

DETAILS OF THE PERIPHERAL ADDRESS CONTROLLER

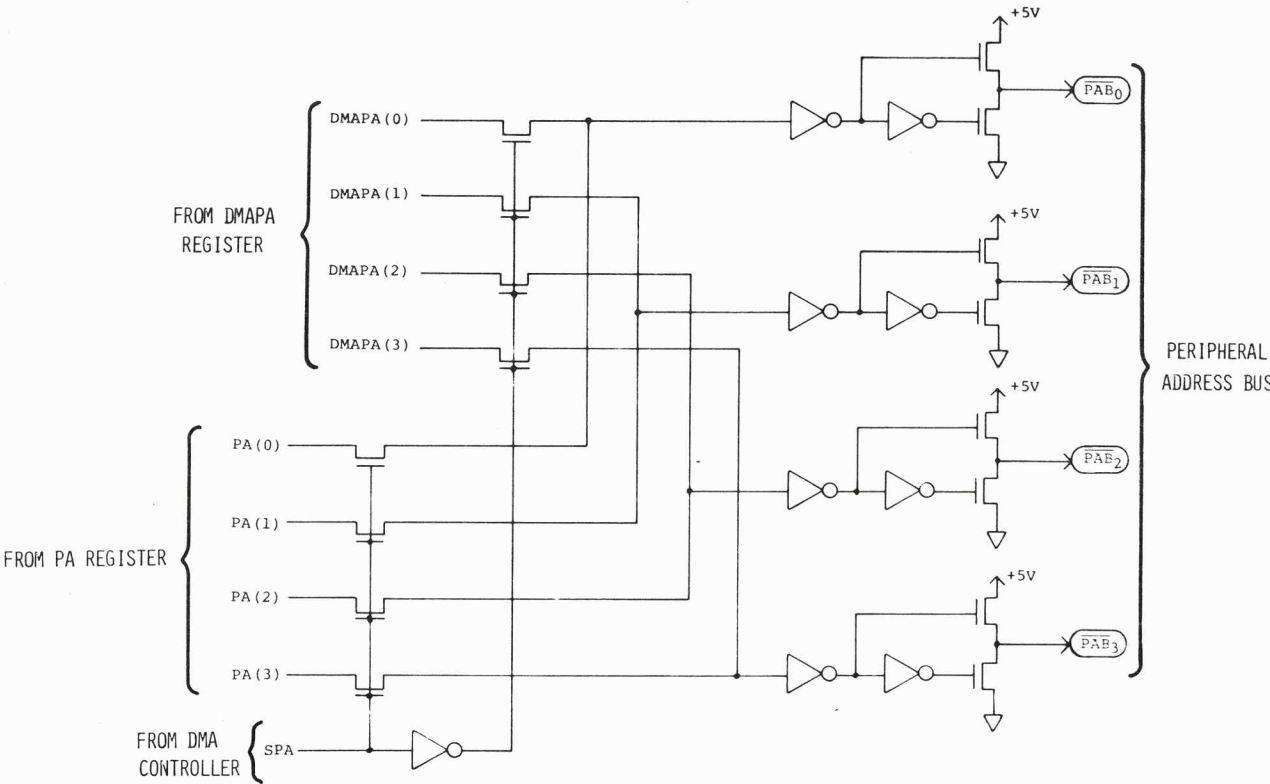
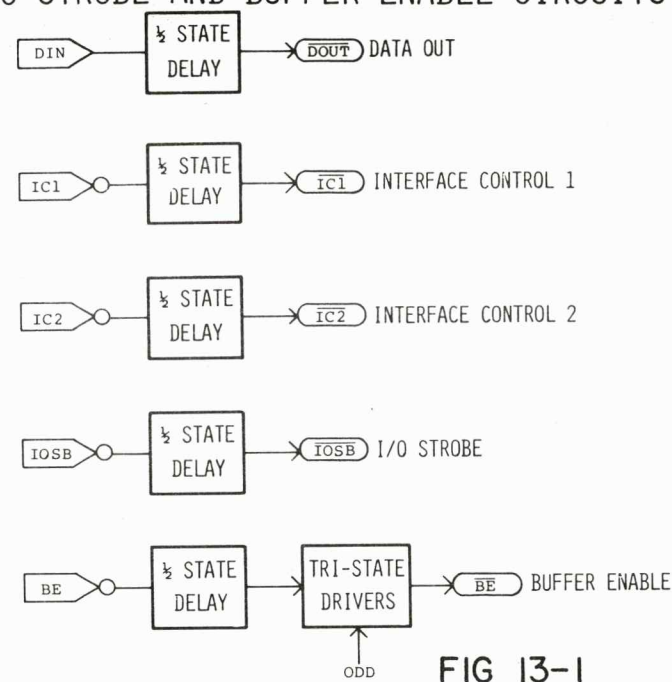


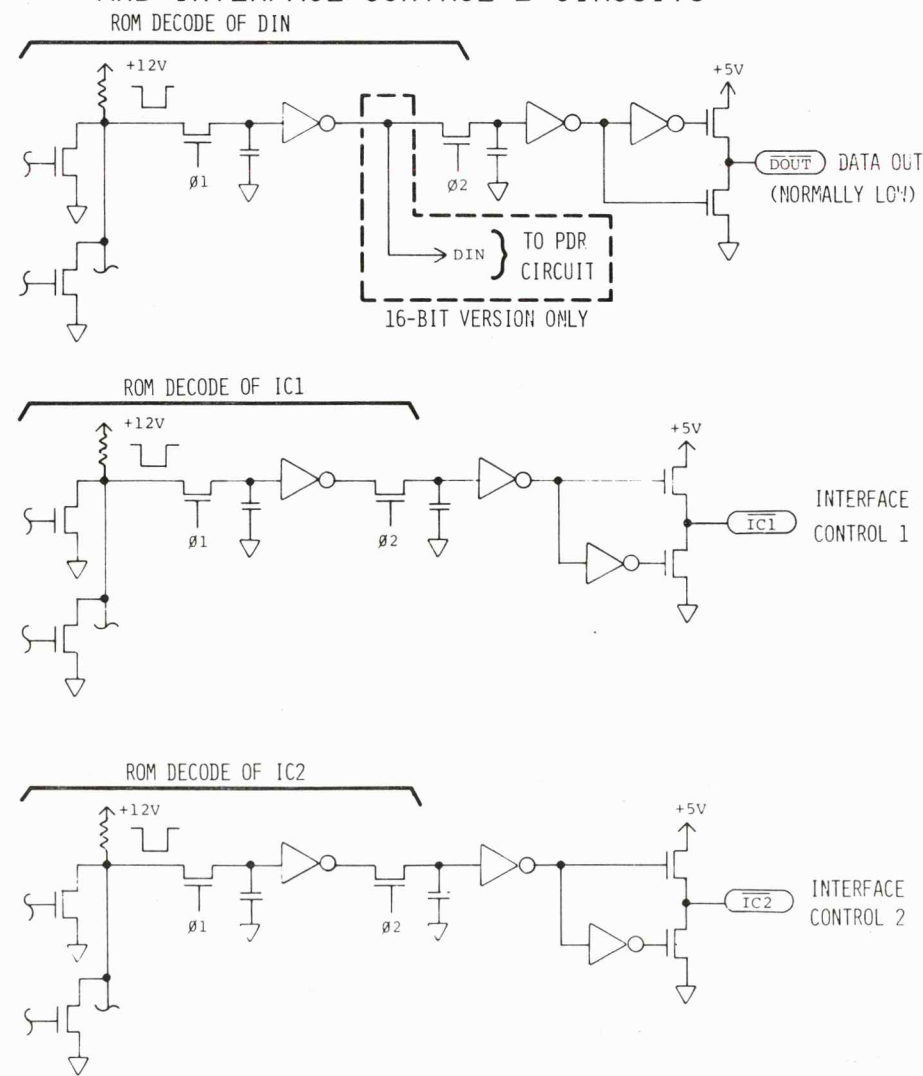
FIG 12-2

Figure 12-2 shows the details of the Peripheral Address Controller. Its operation should be obvious as hell.

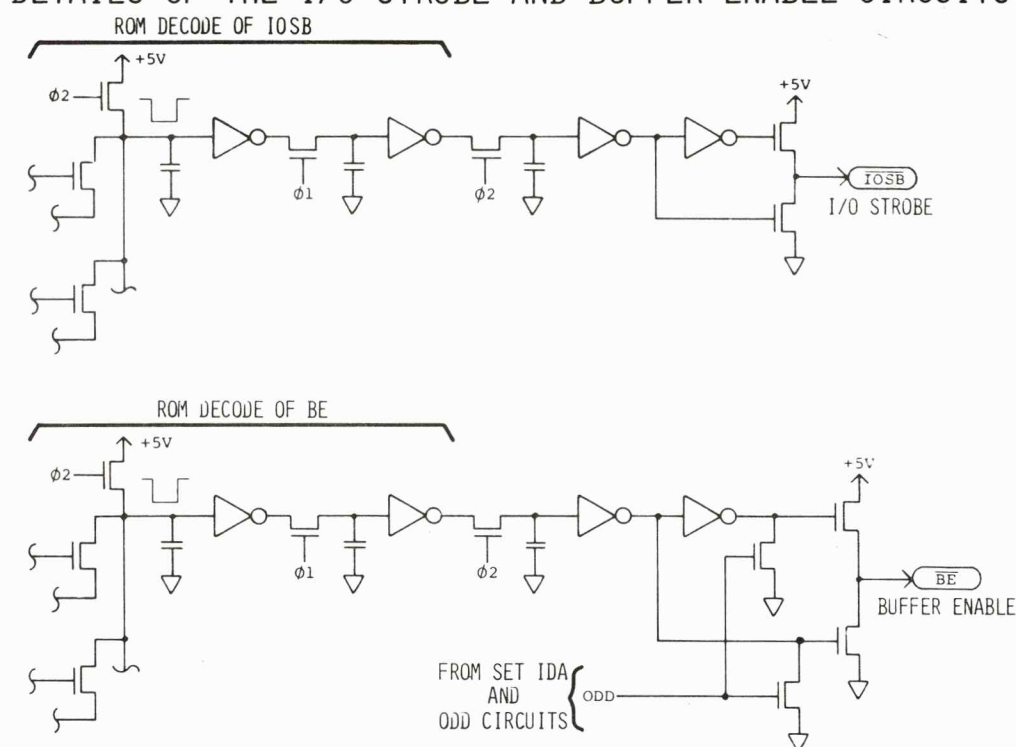
OVERVIEW OF THE DATA OUT, INTERFACE CONTROL, I/O STROBE AND BUFFER ENABLE CIRCUITS



DETAILS OF THE DATA OUT, INTERFACE CONTROL 1, AND INTERFACE CONTROL 2 CIRCUITS



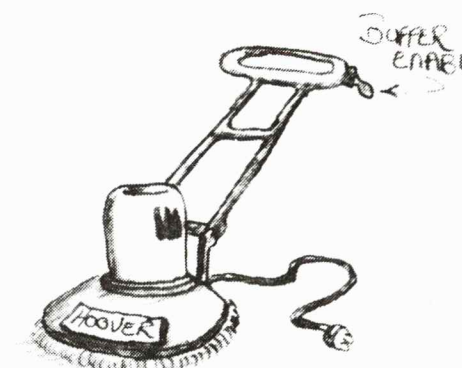
DETAILS OF THE I/O STROBE AND BUFFER ENABLE CIRCUITS



SECTION 13

Figure 13-1 is an overview of the control signals used in communication with peripherals. Data Out indicates the direction of data transfers during an I/O Bus Cycle. Interface Control One and Interface Control Two combine to indicate the nature of any particular I/O Bus Cycle. I/O Strobe indicates the presence of data. Buffer Enable is a signal used to establish the connection between the internal IDC Bus and the external IOD Bus (I/O Data Bus).

Figures 13-2 and 13-3 illustrate the details of these control signals. Observe that in the 16-bit version Data Out contributes to the generation of PDR. Also, observe that Buffer Enable is disabled by ODD.



OVERVIEW OF THE CONTROL ROM

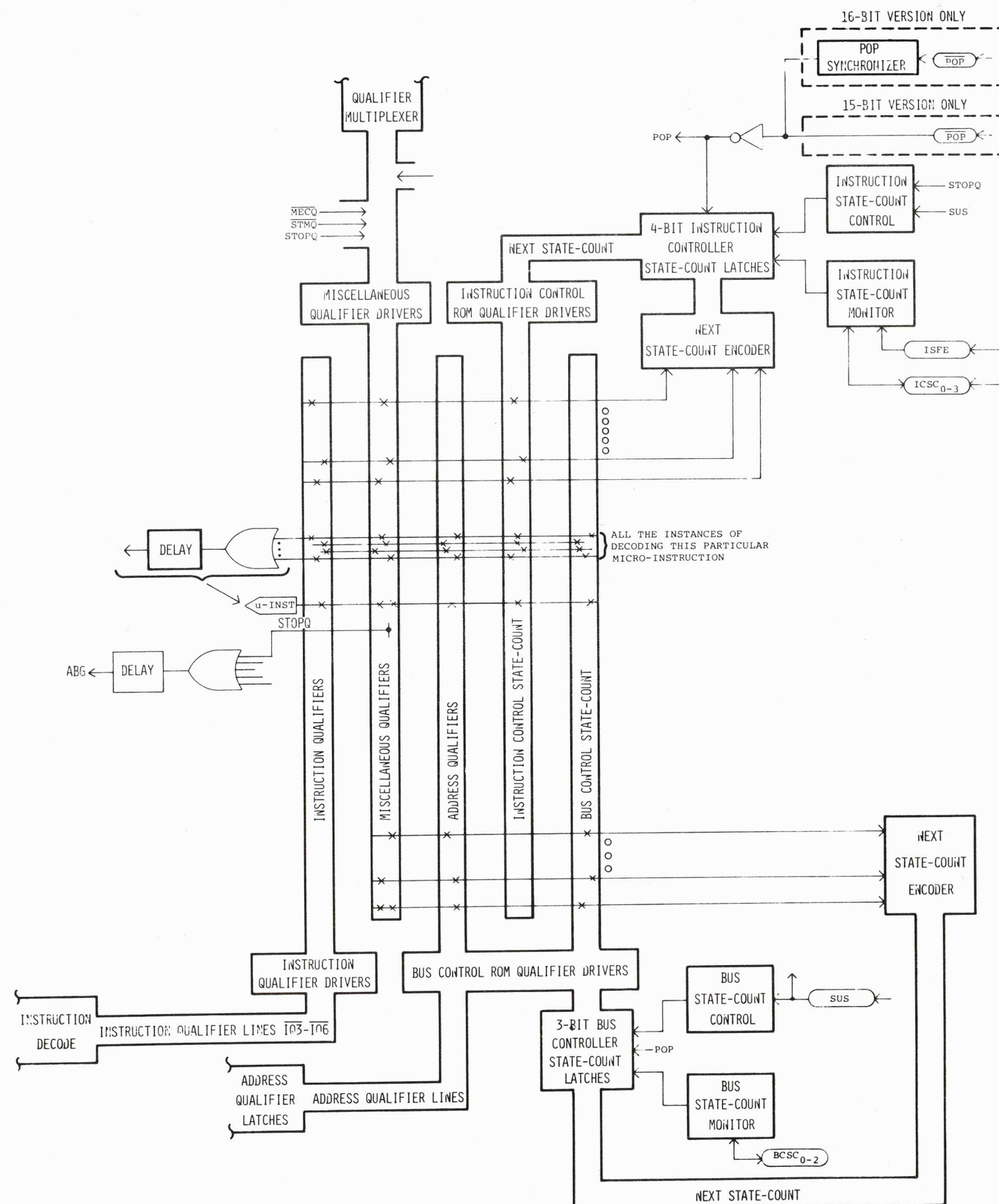


FIG 14-1

SECTION 14

Figure 14-1 is an overview of the main control ROM, which implements the Bus Controller and the Instruction Controller automatic state machines. This amounts to a pooled collection of qualifiers against which a common body of micro-instructions is decoded. Actually, some decoded micro-instructions pertain to only one or the other of the state machines, while some pertain to both.

The Instruction Controller state machine is controlled by a 4-bit state-count in conjunction with instruction qualifier lines from Instruction Decode, and miscellaneous qualifiers supplied, in part, through the Qualifier Multiplexer. The Qualifier Multiplexer allows the same physical qualifier lines in the ROM to represent different logical meanings, depending upon whether DMA or non-DMA operation is in progress.

The State-Counter does not have a native count sequence. Instead, each state decodes the next value for the State-Counter. At turn-on the value of the State-Counter is preset to a predetermined value by POP. The signal STOPQ provides a means to suspend the operation of the Instruction Controller state machine. STOPQ is similar to STP of the BPC. STOPQ halts the Instruction Controller, but not the Bus Controller. The reason for this is that STOPQ is the result of a Bus Request, and a Bus Request generally entails a memory cycle. It is the Bus Controller that manages the IOC's response to memory cycles. To halt the Bus Controller with STOPQ would cause a system lock-up if the memory cycle involved the IOC.

The Instruction State-Count Monitor and the signals ISFE, ICSC-(0-3) and SUS are vestiges of a production test mechanism which does not concern the normal ongoing operation of the IOC.

The Bus Controller state machine is controlled by a 3-bit State-Counter in conjunction with the miscellaneous qualifiers used by the Instruction Controller and address qualifier lines generated by Address Decode. As in the Instruction Con-

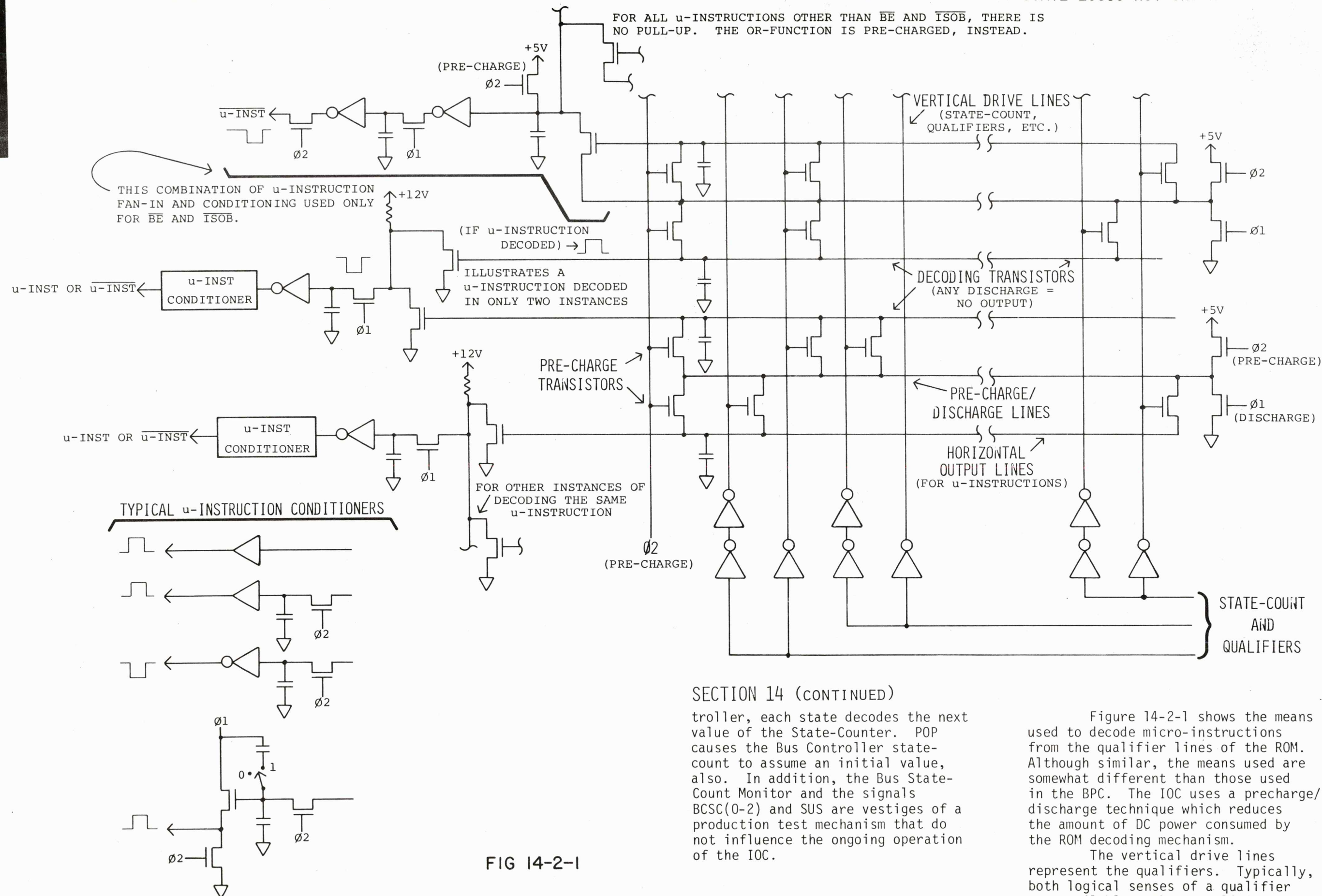


FIG 14-2-1

SECTION 14 (CONTINUED)

troller, each state decodes the next value of the State-Counter. POP causes the Bus Controller state-count to assume an initial value, also. In addition, the Bus State-Count Monitor and the signals BCSC(0-2) and SUS are vestiges of a production test mechanism that do not influence the ongoing operation of the IOC.

Figure 14-2-1 shows the means used to decode micro-instructions from the qualifier lines of the ROM. Although similar, the means used are somewhat different than those used in the BPC. The IOC uses a precharge/discharge technique which reduces the amount of DC power consumed by the ROM decoding mechanism.

The vertical drive lines represent the qualifiers. Typically, both logical senses of a qualifier are available. These qualifier lines will be stable shortly after the

DETAILS OF THE ROM DECODING OF ABG

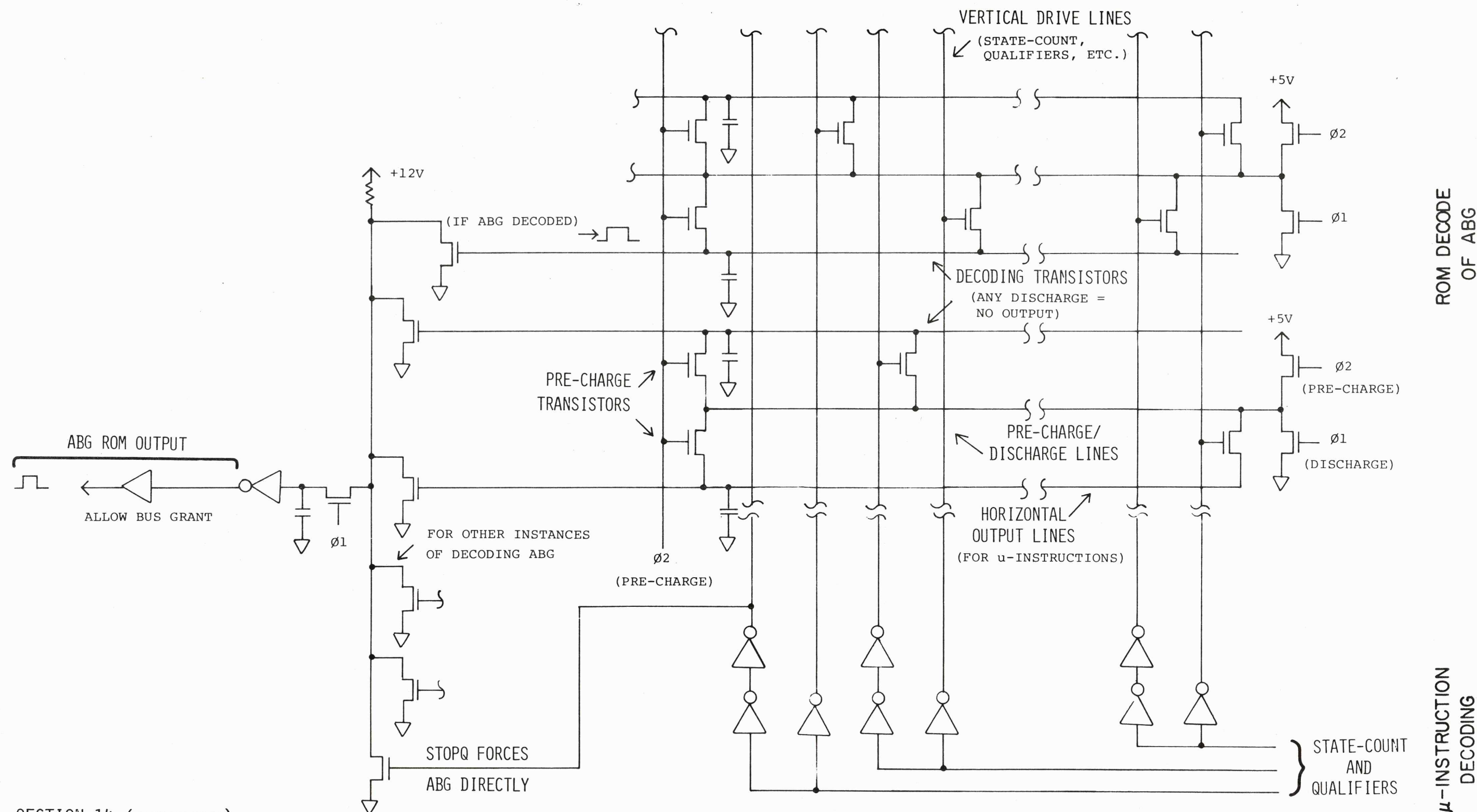


FIG 14-2-2

SECTION 14 (CONTINUED)

onset of phase two. Horizontal output lines come in pairs, and each member of the pair is precharged during phase two by its own precharge transistor. Associated with each pair of horizontal output lines is a single precharge/discharge line. That line is precharged on phase two and discharged on phase one. A

horizontal output line is logically true (decoded) when, during phase one, values of the qualifiers are such that the output line is *not* connected to its discharge line. Under those circumstances the pre-charge on the horizontal output line remains, and is not drained away by the phase one discharge.

The drawing also shows several typical ways used to collect the decoded micro-instructions, condition them and send them to their destinations.

Figure 14-2-2 illustrates the decoding of the micro-instruction Allow Bus Grant (ABG). The actual decoding of this micro-instruction is as previously described for other micro-instructions. The significance of this drawing is that it shows that STOPQ generates an ABG. STOPQ occurs during DMA operation.

106

DETAILS OF THE NEXT STATE-COUNT LOGIC AND STATE-COUNT LATCHES FOR INSTRUCTION AND BUS CONTROL ROMS

DETAILS OF THE POP CIRCUIT, CONTROL AND STATE COUNT MONITOR FOR THE BUS CONTROL ROM.

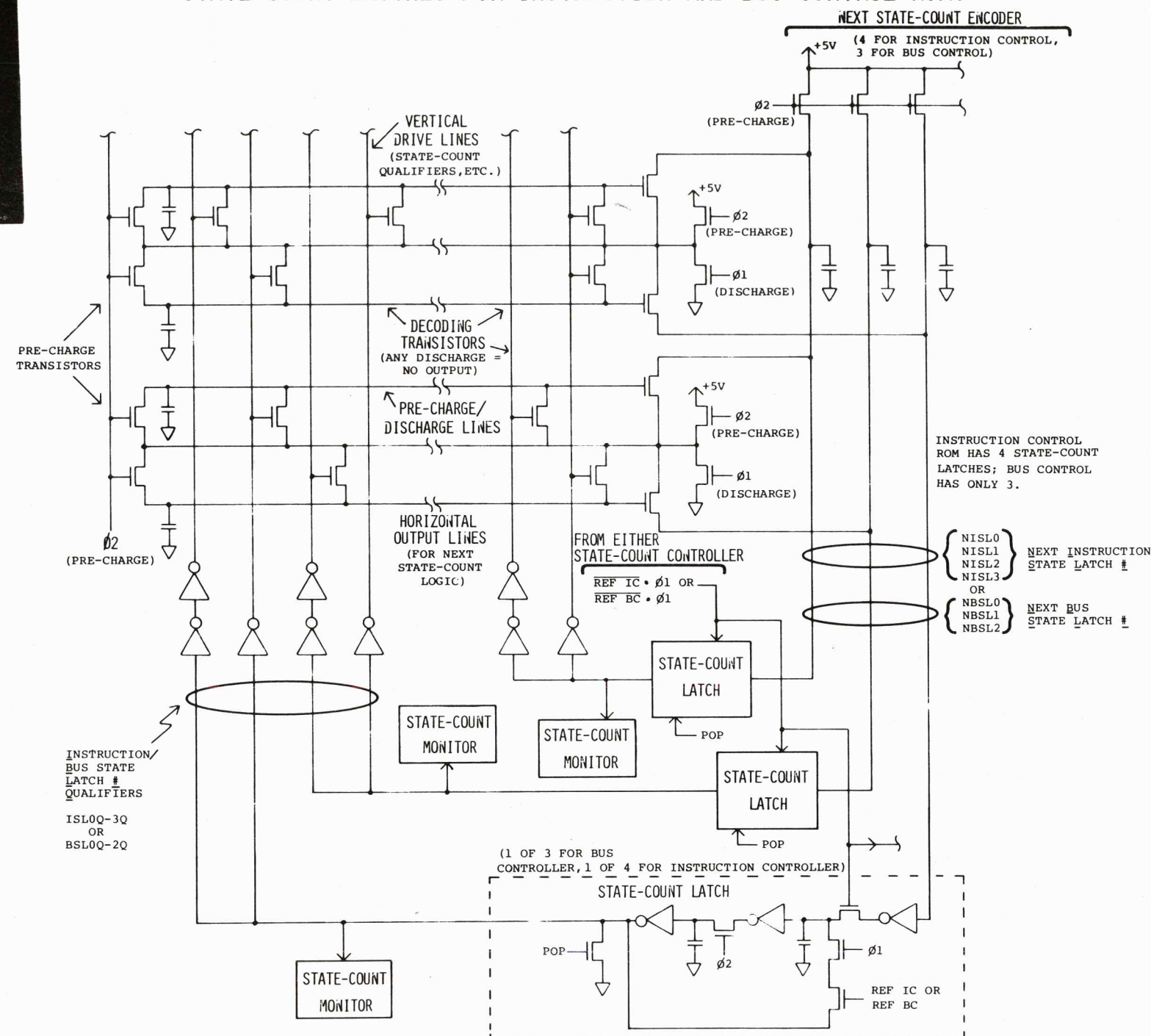


FIG 14-3

SECTION 14 (CONTINUED)

Figure 14-3 illustrates the State-Counters used by the Instruction Controller and the Bus Controller state-machines. Observe how POP forces an initial value for the State-Counter. The signals REF IC and REF BC are each generated as a result of SUS. Since that mechanism is unused during the operation of the IOC, both REF IC and REF BC can

always be assumed to be false. The State-Count Latches are set according to next state-count information decoded from the ROM itself, as described below.

"Next State-Count" micro-instructions are decoded from the ROM in the same manner as are other micro-instructions. Unlike as in the BPC, however, the decoding of a

single micro-instruction does not represent an entire state-count. Instead, each micro-instruction concerns itself only with a single bit of the associated State-Counter. In general, several micro-instructions must be decoded to determine the next state-count. Observe how each horizontal output line goes to the gate of a transistor connected to

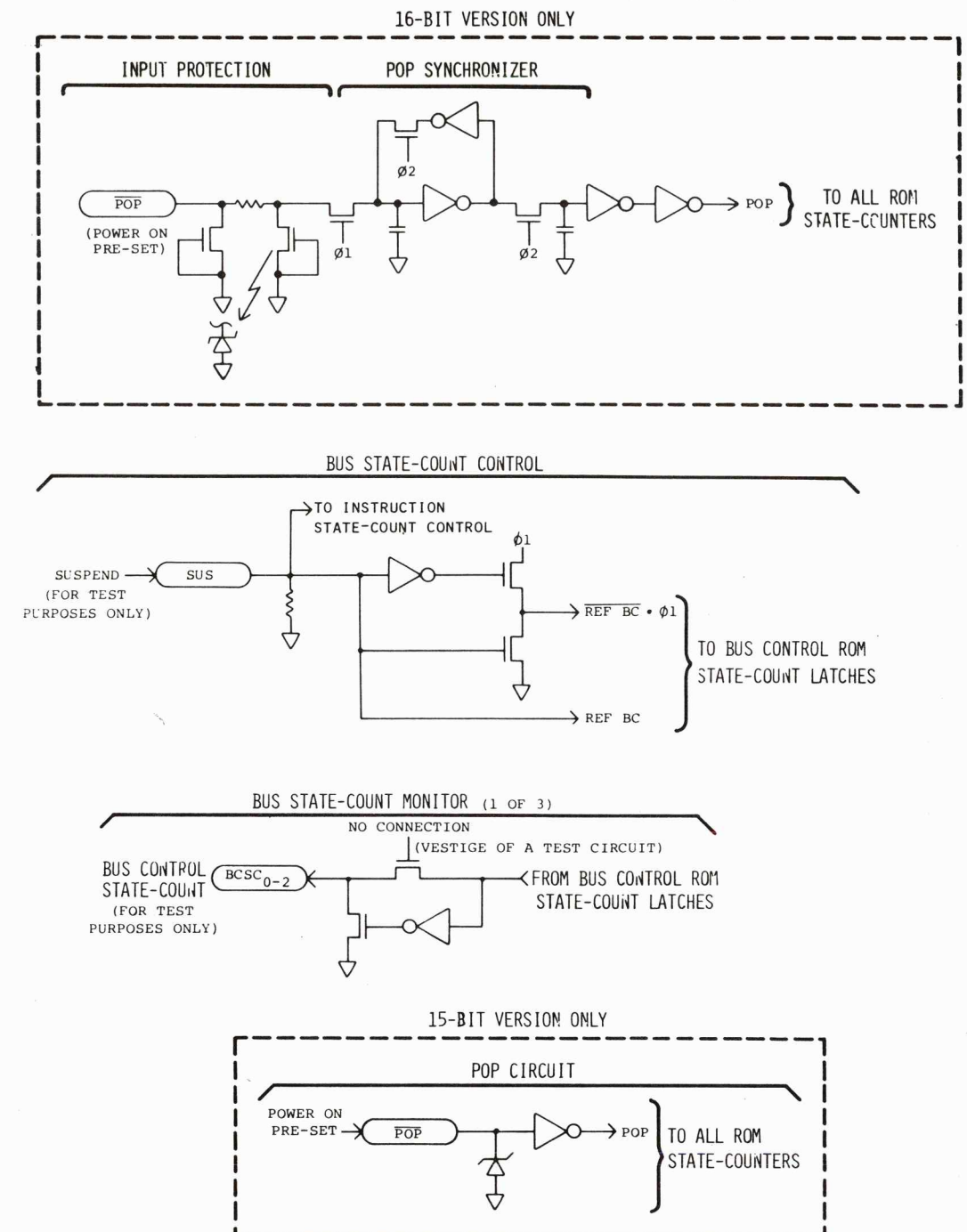


FIG 14-4

the precharge/discharge line. A logically selected horizontal output line will turn on that transistor, and during phase one the pre-charged line leading to the associated State-Count Latch will be grounded.

Figure 14-4 shows the details of the POP circuitry for both versions, as well as the details of

DETAILS OF STATE-COUNT CONTROL AND THE STATE-COUNT MONITOR FOR THE INSTRUCTION CONTROL ROM.

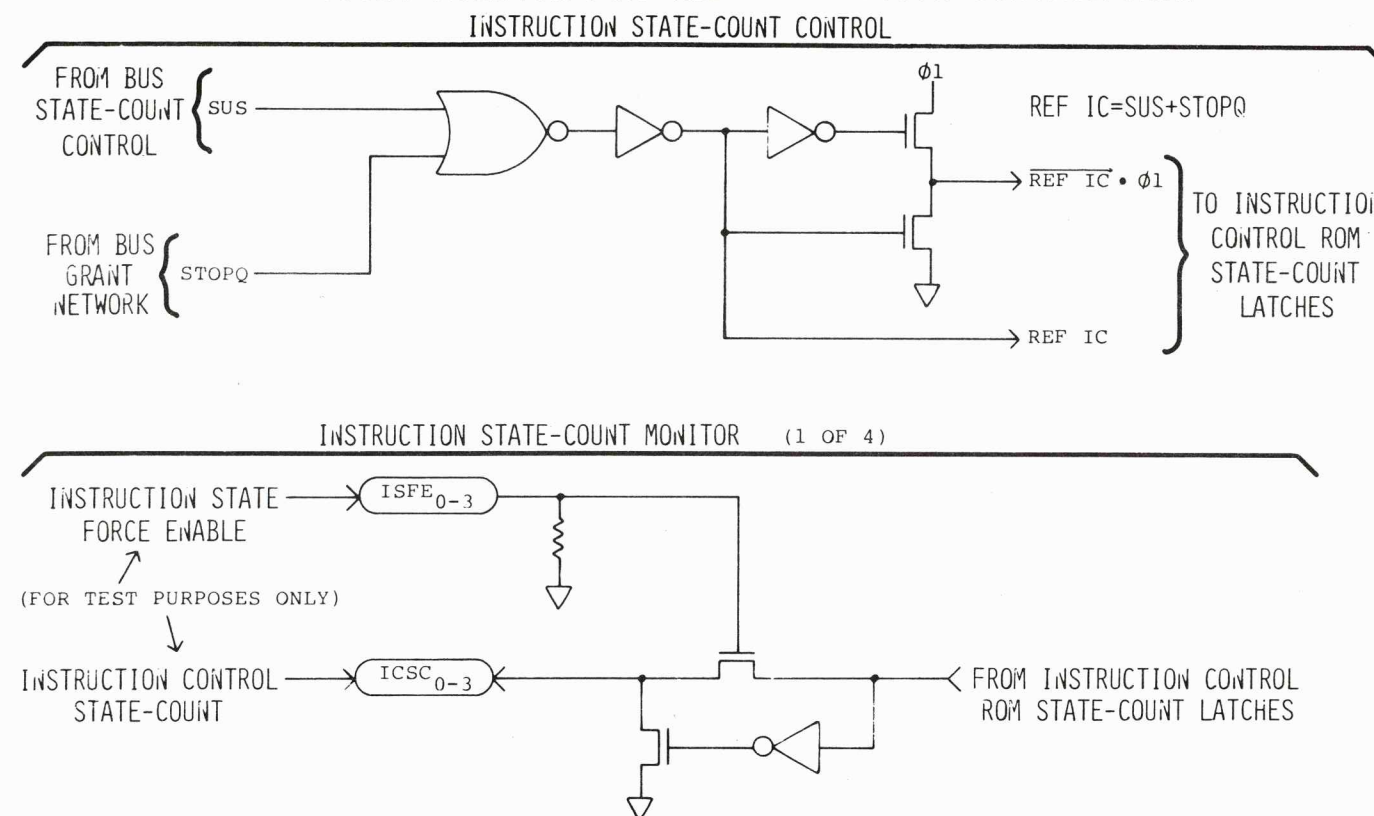


FIG 14-5

SECTION 14 (CONTINUED)

certain control circuitry used in developmental and production testing. The control circuitry is of no functional use during normal IOC operation.

With regard to the 16-bit version's POP Synchronizer: it's alleged not to work. No one has analyzed it to find out just what's wrong with it.

Figure 14-5 illustrates how STOPQ is used to suspend the operation of the Instruction Controller State-Counter. It also illustrates some

unused circuitry associated with that state machine.

OVERVIEW OF THE QUALIFIER MULTIPLEXER AND SYNC CIRCUIT

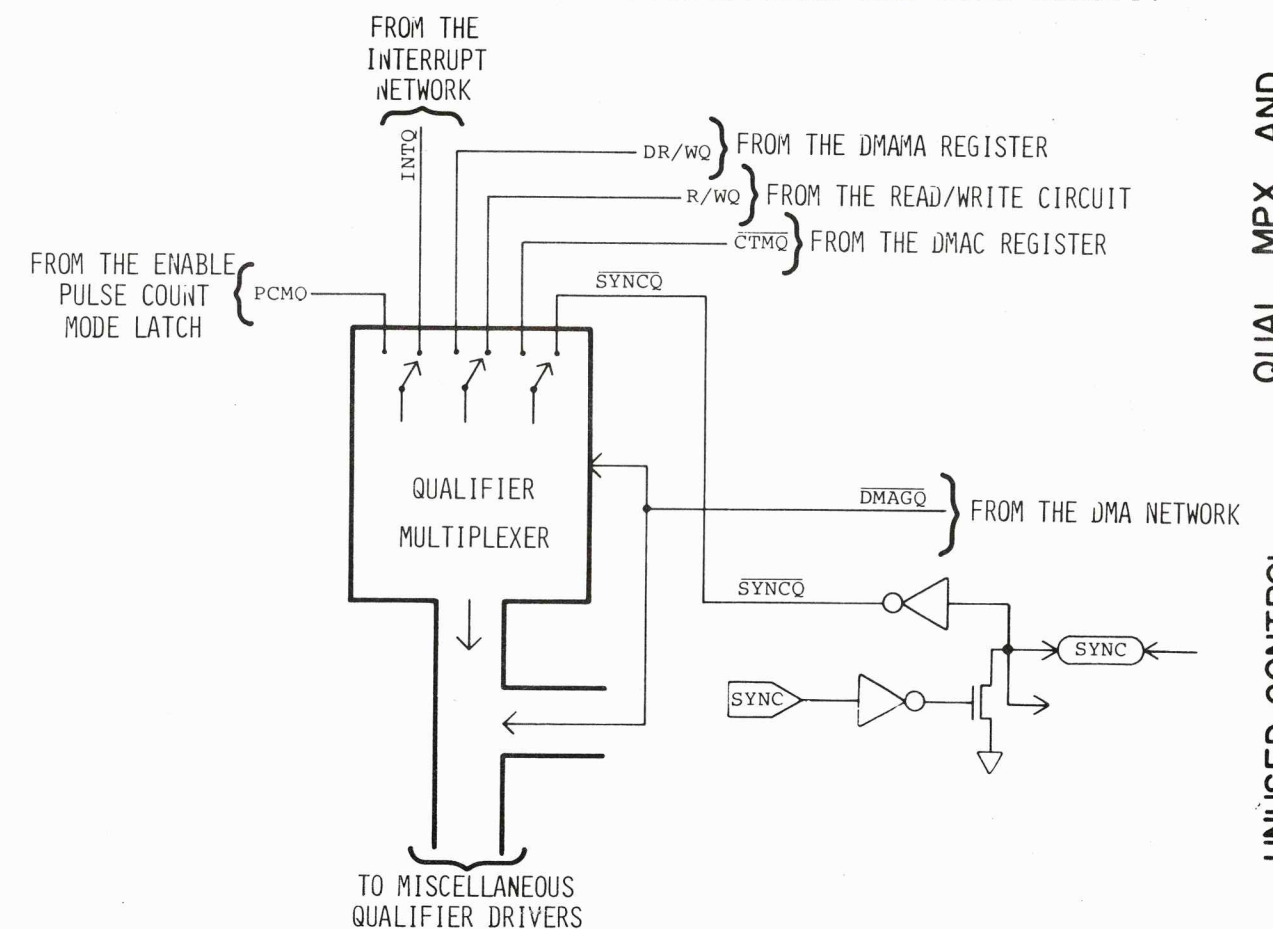


FIG 15-1

SECTION 15

Figure 15-1 is an overview of the Qualifier Multiplexer and of the SYNC circuitry. The purpose of the Qualifier Multiplexer is to make six qualifiers into three. The Qualifier Multiplexer chooses between DMA related and non-DMA related qualifiers, on the basis of the signal DMAGQ (DMA Grant Qualifier). That signal originates in the DMA Network and represents either the DMA or Pulse Count Mode of operation on the one hand, and non-DMA related activity on the other.

Observe that the SYNC line

of the IOC is both a mouth and an ear. Notice also that SYNCQ is a qualifier available to the ROM during non-DMA operation. (This is sufficient because SYNCQ is used *only by the Instruction Controller* during the execution of IOC related machine-instructions. The operation of the Instruction Controller is suspended by STOPQ during DMA operations. This suspension of the Instruction Controller state-machine is the basis for the ability of the Qualifier Multiplexer to change the definition of qualifier lines running to the ROM).

QUAL. MPX. AND
SYNC CIRCUIT
OVERVIEW

UNUSED CONTROL
CIRCUITS

POP CIRCUIT

NEXT STATE LOGIC
AND STATE-COUNT
LATCHES

DETAILS OF THE SYNC OUTPUT CIRCUIT

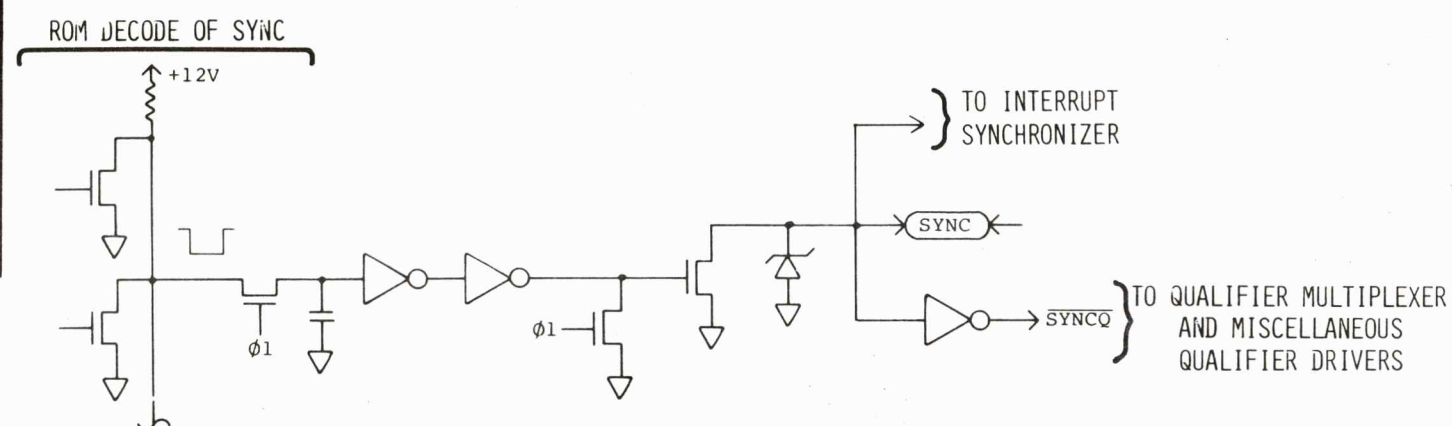


FIG 15-2

SECTION 15 (CONTINUED)

Figure 15-2 illustrates the details of the ROM decoding of SYNC by the IOC. Observe that the IOC can ground SYNC only during phase one. Recall however, that just because the IOC has decoded SYNC, this does not mean that SYNC will be true during phase one. Some other (external) agency may keep SYNC grounded.

DETAILS OF THE QUALIFIER MULTIPLEXER

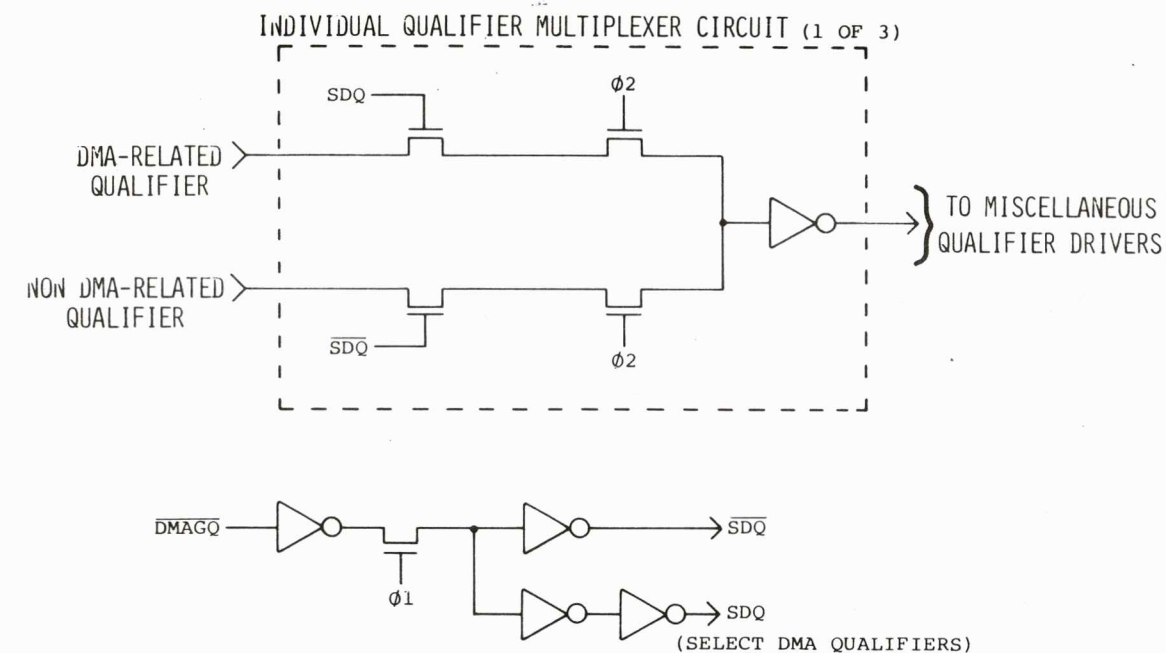


FIG 15-3

Figure 15-3 illustrates the actual mechanism of the Qualifier Multiplexer. The phase two transfer gates in series with the qualifiers ensure that the actual physical qualifier line sent to the ROM is stable during phase one.

OVERVIEW OF ADDRESS DECODE AND THE ADDRESS QUALIFIER LATCHES

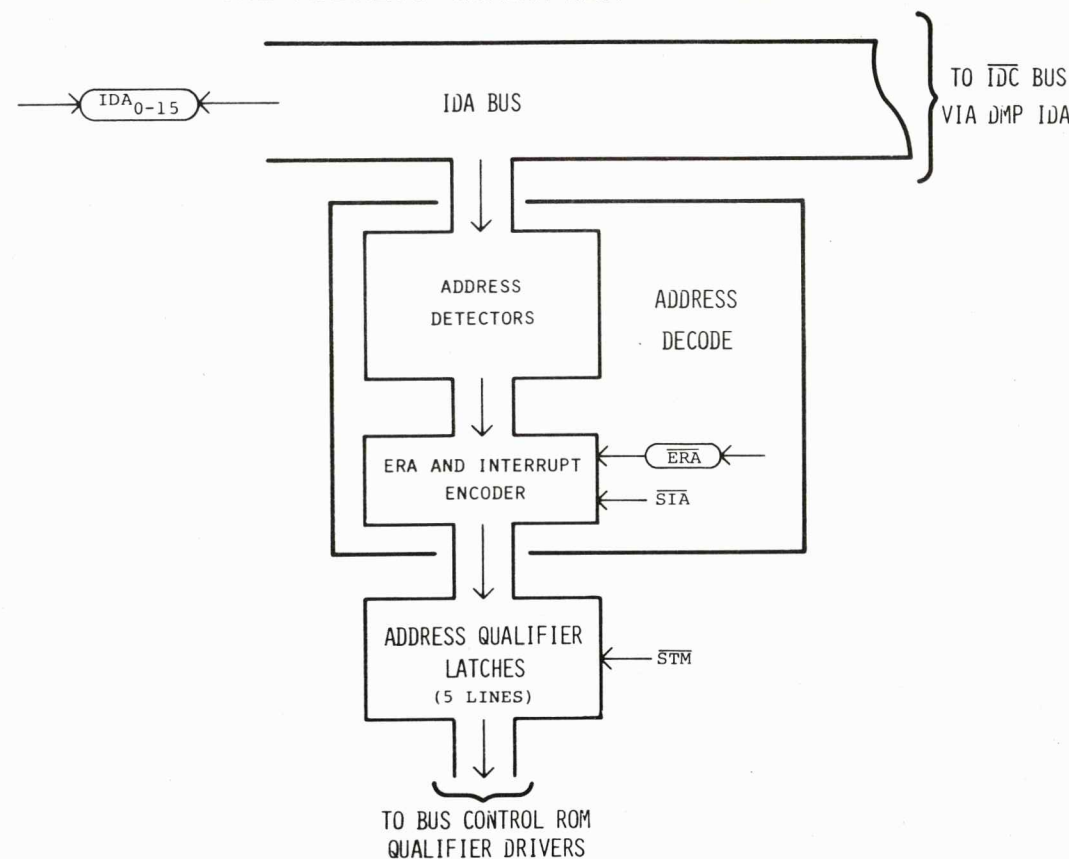


FIG 16-1

DETAILS OF THE ADDRESS DETECTORS AND OF THE ERA AND INTERRUPT ENCODER.

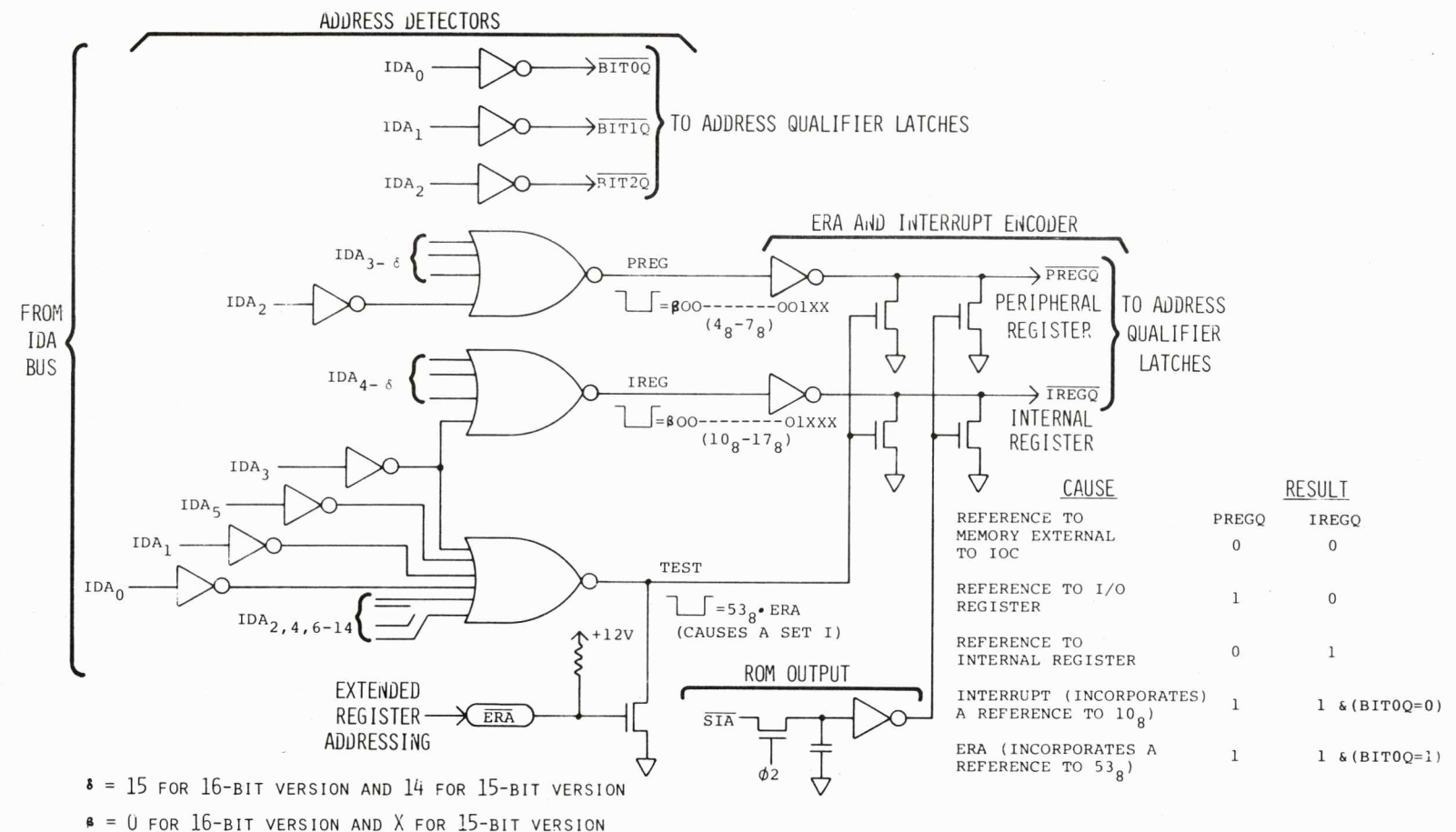


FIG 16-2

SECTION 16

Figure 16-1 is an overview of Address Decode and the Address Qualifier Latches. The Address Detectors are sensitive to the various addresses to which the IOC must respond. This concept is not as simple as it was in the BPC, however. Some addresses represent actual registers that physically exist within the IOC, while other addresses represent more abstract concepts. An example is a reference to registers 4-7, which is interpreted by the IOC as a command to undertake communication with the currently addressed peripheral. The occurrence of ERA Mode addressing and of interrupts add additional complexity to the problem of address identification.

At any rate, the results of the address detection process are latched and sent to the Bus Control ROM as qualifiers. A derivative of Start Memory is used to enable the latches.

Figure 16-2 shows the details of the Address Detectors and of the ERA and Interrupt Encoder. The Address Detectors consist of gating that is organized to respond to selected categories of addresses. IREG represents the occurrence of an address for a register internal to the IOC. PREG represents the occurrence of an address of a pseudo-register intended to designate peripheral activity (I/O). TEST is associated with addresses invoked during ERA Mode addressing.

Within a given category of addresses, the bottom three bits are represented for what they are, and serve in the obvious manner to differentiate a location from its neighbor. PREG and IREG are formed into qualifiers sent to the ROM. These qualifiers can have four possible combinations of values. Memory cycles that reference memory external to the IOC, or that re-

ference either peripheral registers or internal registers of the IOC, account for only three of the four possibilities. The purpose of the ERA and Interrupt Encoder is to exploit the remaining possibility by causing it to represent either ERA Mode addressing or the occurrence of an interrupt. Both these cases cause a certain pattern of $\overline{\text{PREGQ}}$ and $\overline{\text{IREGQ}}$, as shown on the drawing. The two cases are further distinguished between each other by the value of bit 0 on the IDA Bus. That occurs as follows. The only instance of ERA Mode addressing for the IOC is a reference to 53₈. This causes a SET I which may be used for test purposes. On the other hand, the situation concerning interrupt involves the BPC's interrogation of the IV register. Its address is 10₈. This difference in associated address accounts for the difference in bit 0 of the IDA Bus.

Observe how both ERA Mode activity and the occurrence of an interrupt force the 4th possibility of the two qualifiers PREGQ and IREGQ. It's easy to understand where the ERA came from; some external agency grounded ERA. Interrupt is not so easy to understand. In a nut shell it's this way. The Interrupts Controller has decided to allow a requested interrupt. That generates the qualifier INTQ. Now, interrupts can occur only during instruction fetches. During an instruction fetch the Instruction Controller state-machine is in an instruction fetch loop. At the conclusion of the instruction fetch the occurrence of INTQ causes that state-machine to branch to a segment that helps generate the interrupt vector, in preparation for the BPC's interrogation of IV. It is that segment of flow charting that generated the SIA which caused the ERA and Inter-

DETAILS OF THE ADDRESS QUALIFIER LATCHES

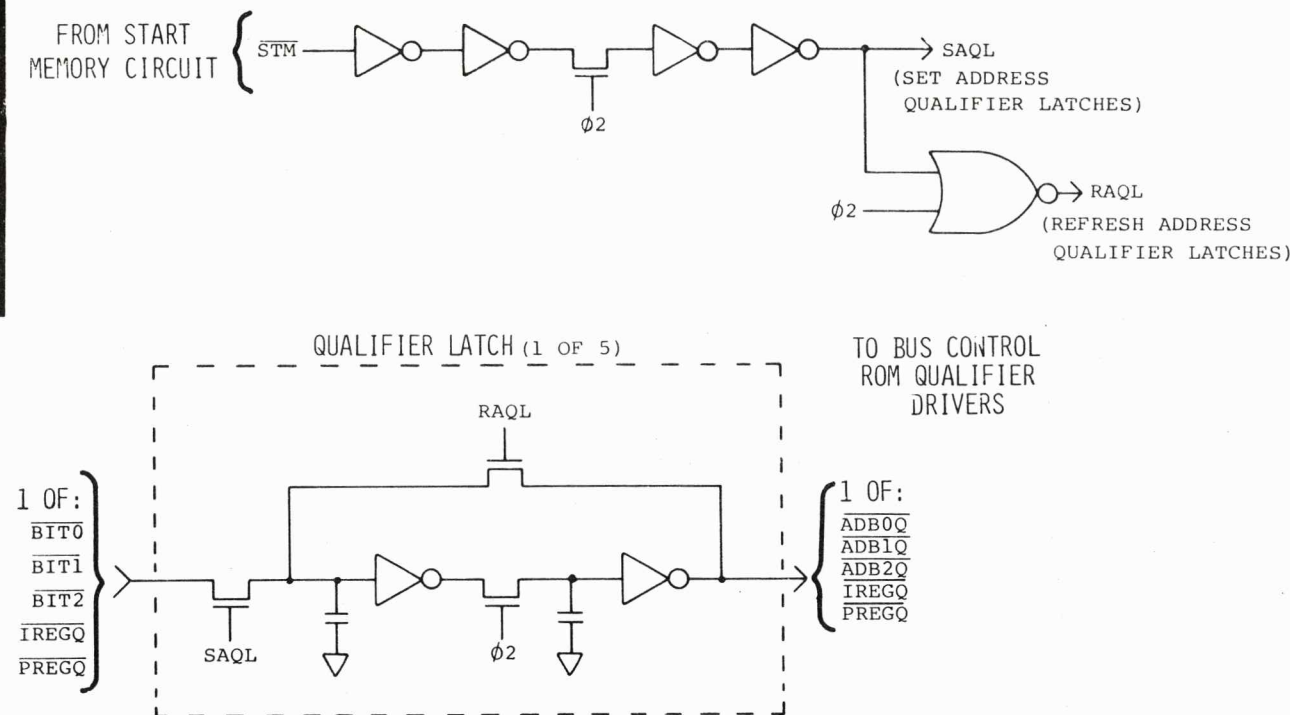


FIG 16-3

SECTION 16 (CONTINUED)

rupt Encoder to force the 4th possibility.

Well, so what? Why bother to force PREGQ and IREGQ to a particular state simply because an interrupt is in progress? The answer is this. *The BPC's interrogation of IV is not a simple memory cycle.* Such an interrogation involves the performance of an interrupt poll. The Bus Controller state machine responds to memory cycles as well as performing actual interrupt polls. Now, the Bus Controller state machine is set in motion by memory cycles referencing the IOC. A reference to register 10₈ during an interrupt is a different animal than a mere reference to register 10₈ for purposes of setting it or interrogating it. That is, a simple read or write memory cycle directed to register 10₈ is just that, a

memory cycle. However, the kind of operation that has to occur when the BPC does its JSM 10₈, I is *not* a mere memory cycle, and is handled by a very special segment of flow charting in the Bus Controller ASM chart. The special values of PREGQ and IREGQ, which are qualifiers for the Bus Controller state machine, see to it that that special segment of flow charting in the Bus Controller is reached.

The differences between the 15 and 16-bit versions concerns merely the extra bit in the Address Detectors.

Figure 16-3 illustrates the details of the Address Qualifier Latches. Observe how a slightly delayed STM is used to generate signals to set or refresh the latches.

OVERVIEW OF THE READ/WRITE, START MEMORY, AND SYNCHRONIZED MEMORY COMPLETE CIRCUITS.

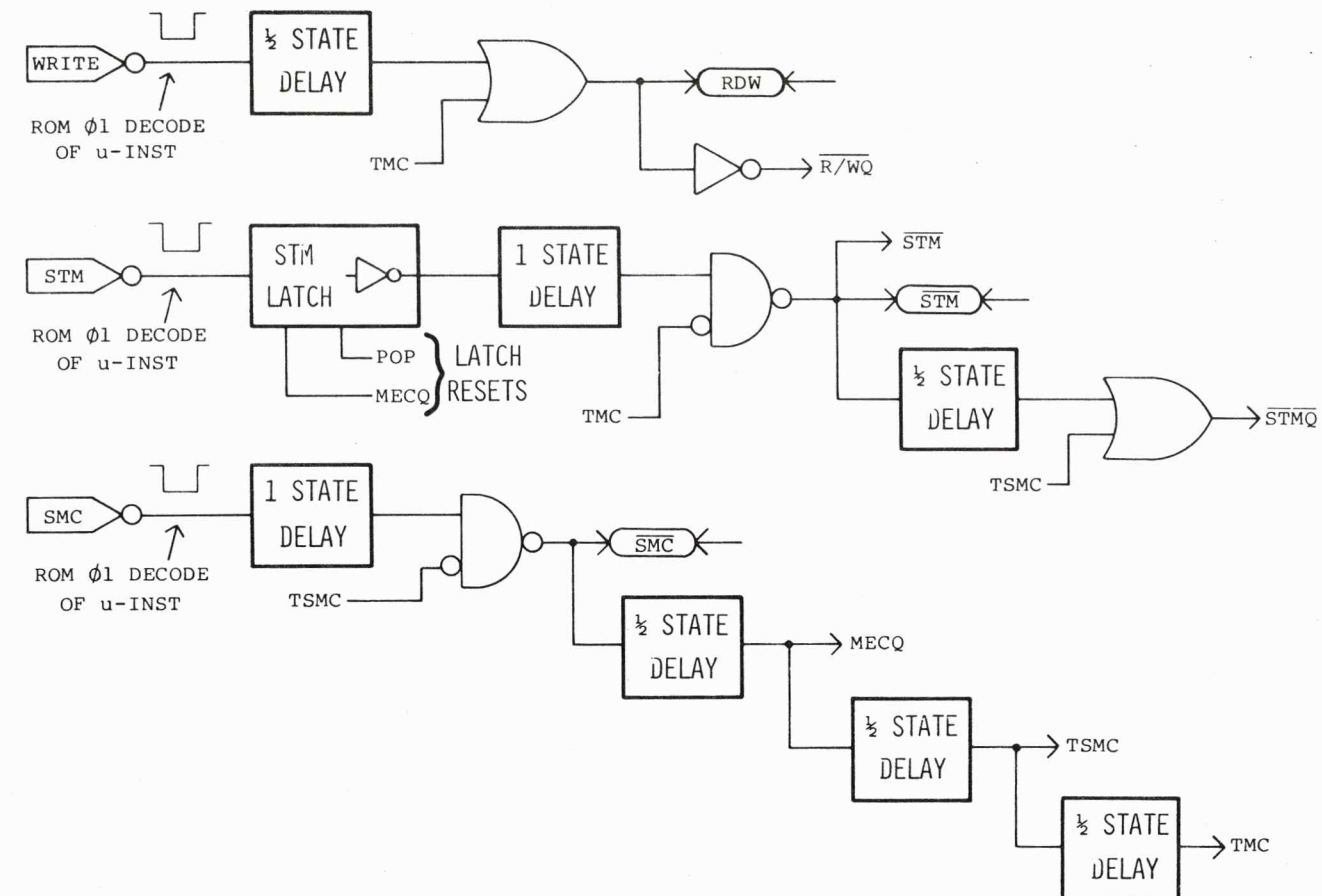


FIG 17-1

SECTION 17

Figure 17-1 is an overview of the Read/Write, Start Memory and Synchronized Memory Complete circuitry. Observe that Read/Write is controlled by an unlatched ROM output (WRITE) which must be given for the duration of the write operation. Now, it is undesirable to keep RDW grounded beyond the time it takes to recognize the occurrence of a Synchronized Memory Complete. However, if one were to rely upon the change in qualifiers controlled by SMC, and their subsequent effect on the decoding of WRITE, it would take too long for RDW to go high. The solution is to use TMC to force RDW high.

As in the BPC, the ROM decode of Start Memory sets a latch. Thus, Start Memory need be decoded from the ROM only once. The latch is reset at power-on and by Memory Complete. After a suitable delay, a decoded

Start Memory is issued to the Bus. Because of the delay between the latch and the actual Start Memory line, Memory Complete's resetting of the latch does not reset the actual Start Memory line quick enough. Once again, TMC is used to remedy this.

The state of the Start Memory line is turned into a qualifier (STMQ) for use by the Bus Controller. This qualifier is also given a fast reset by TSMC.

When issued from the ROM, the SMC micro-instruction endures a one-state delay before driving the SMC line. A series of cascaded half-state delays produce the qualifier MECQ and the two reset signals TSMC and TMC.

DETAILS OF THE READ/WRITE CIRCUIT

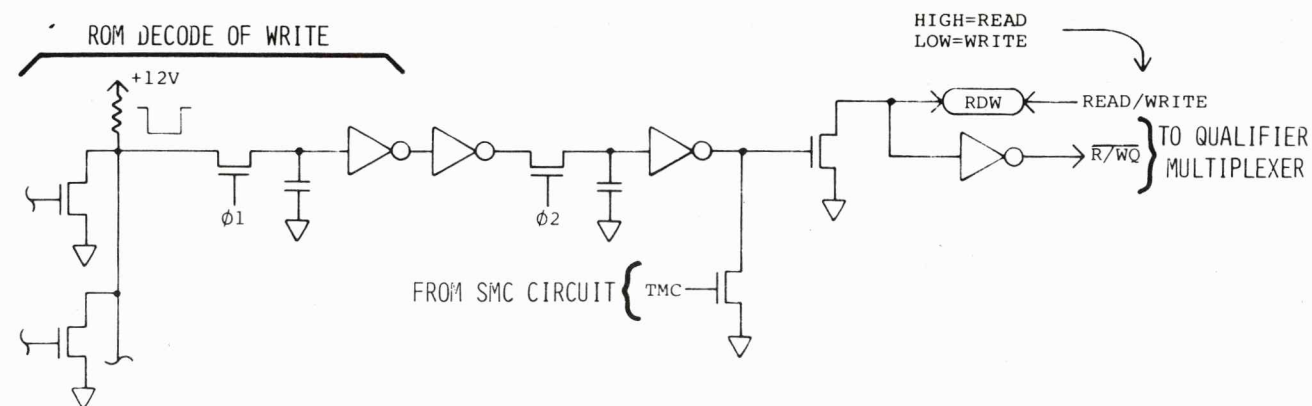


FIG 17-2

DETAILS OF THE START MEMORY AND SYNCHRONIZED MEMORY COMPLETE CIRCUITS

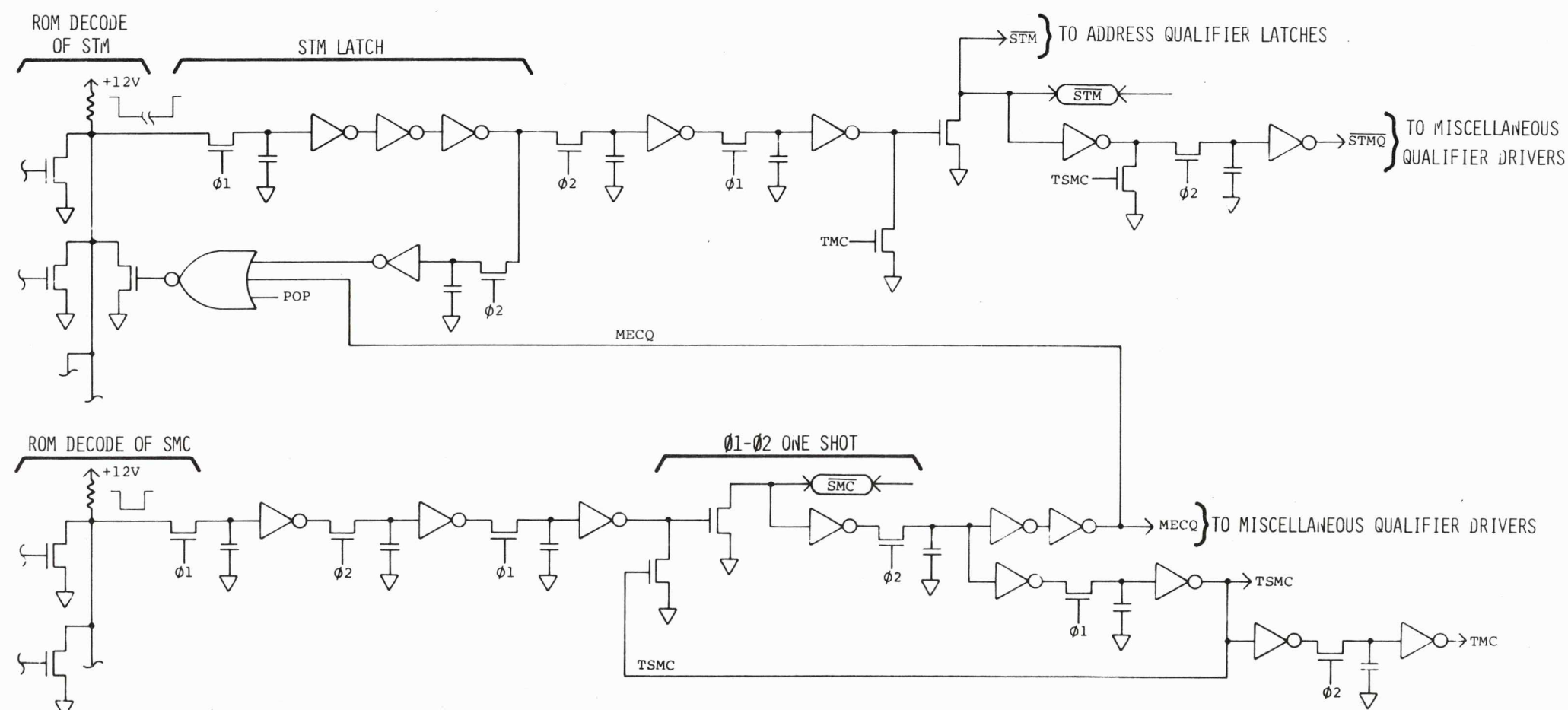


FIG 17-3

SECTION 17 (CONTINUED)

Figures 17-2 and 17-3 show the details of the previously mentioned circuitry.

STM & SMC

RDW

RDW, STM & SMC
OVERVIEW

ADDRESS QUALIFIERS
LATCHES

OVERVIEW OF THE INTERRUPT REQUEST, DMA REQUEST, AND PULSE COUNT MODE LATCHES

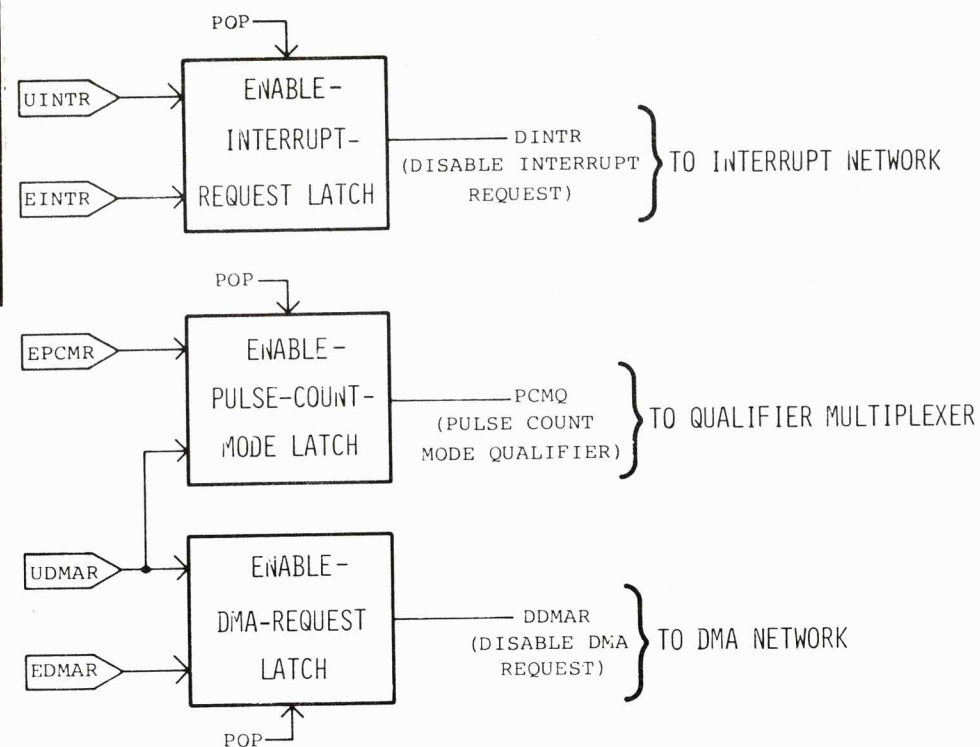


FIG 18-1

SECTION 18

Figure 18-1 is an overview of the Enable Interrupt Request Latch, Enable Pulse Count Mode Latch and the Enable DMA Request Latch. The Enable Pulse Count Mode Latch generates a genuine qualifier used by the ROM. The other two generate control signals that are used in a qualifier-like manner by the Interrupt Network and the DMA Network. Observe that each latch is reset at power-on.

The Enable Interrupt Request Latch represents whether the Interrupt System is enabled or disabled. Those conditions can be put into effect by the machine-instructions EIR and DIR, respectively. Two signals control the Enable Interrupt Request Latch. These are the ROM outputs UINTR and EINTR. UINTR stands for Update Interrupt Request. When UINTR is given the latch will then be set or cleared, based on the value of EINTR, which stands for Enable Interrupt Request. The purpose of the other two latches is to represent the existence or absence of each of the DMA and Pulse Count Modes. They are controlled by the DMA, PCM, and DDR machine-instructions.

These other two latches share a common update signal. That signal is UDMAR, which stands for Update DMA Request. Each of the two latches has its own discrete enable signal. These are EPCMR and EDMAR, as shown. It's desirable for these two latches to share a common update signal for the following reason. The establishment of the DMA or Pulse Count Mode always involves simultaneous setting of *both* latches to some predetermined combinations.

DETAILS OF THE ENABLE INTERRUPT REQUEST LATCH, ENABLE PULSE COUNT MODE AND ENABLE DMA REQUEST LATCHES

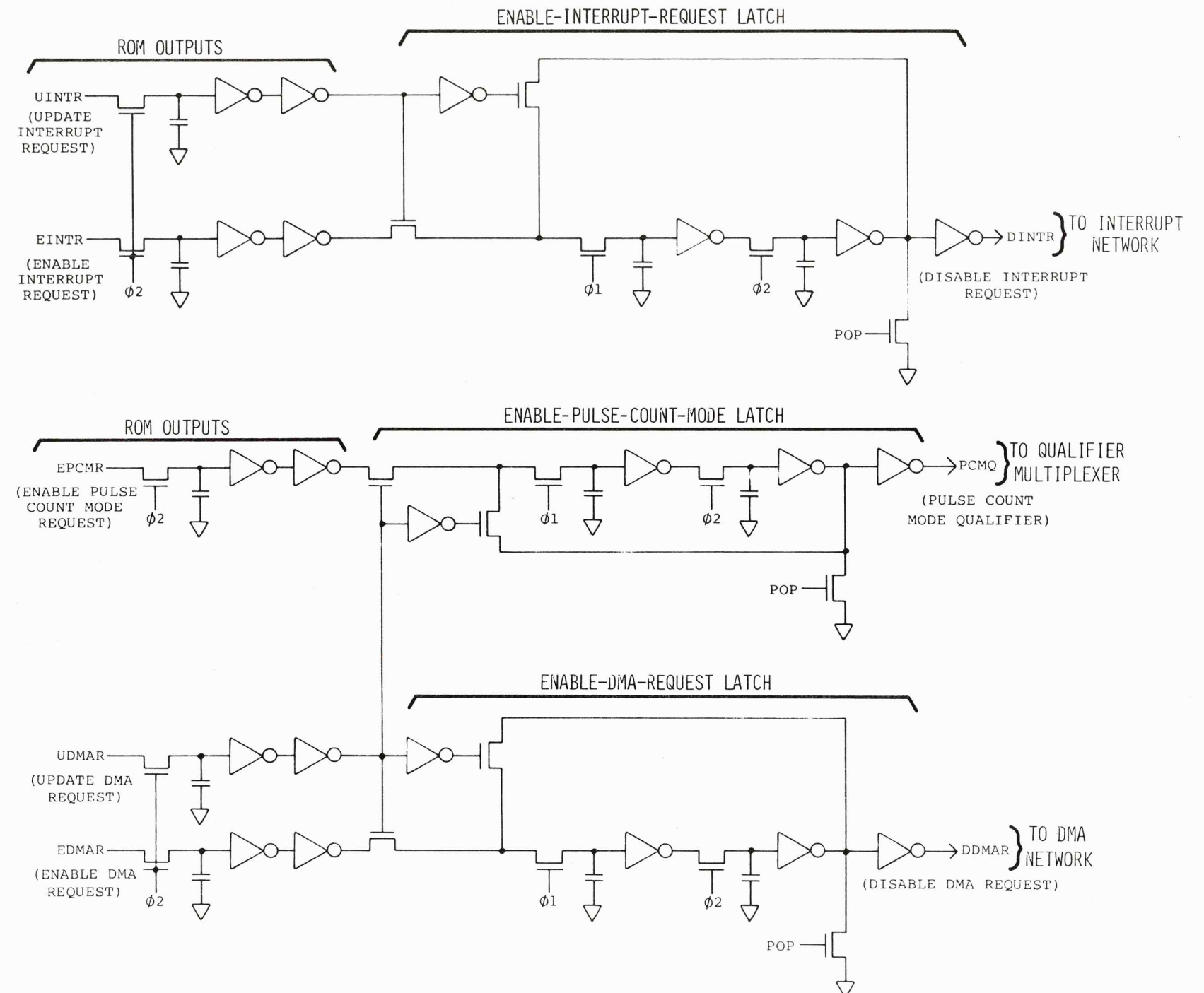


FIG 18-2

The DMA Mode is established by setting the Enable DMA Request Latch and clearing the Enable Pulse Count Mode Latch. The Pulse Count Mode is established by setting both latches. (After all, the Pulse Count Mode requires the use of the DMA Request line. To not set the DMA latch, as we'll later see, would render the DMA Request Line inoperative.) Under this scheme the meanings of the two latches amount to this. The Enable DMA Request

Latch controls whether the DMA Request line is operative, and the Enable Pulse Count Mode Latch determines whether a use of the DMA Request line results in a DMA operation or in a Pulse Count Mode operation.

Figure 18-2 shows the details of the above-mentioned latches.

OVERVIEW OF THE INTERRUPT CONTROLLER

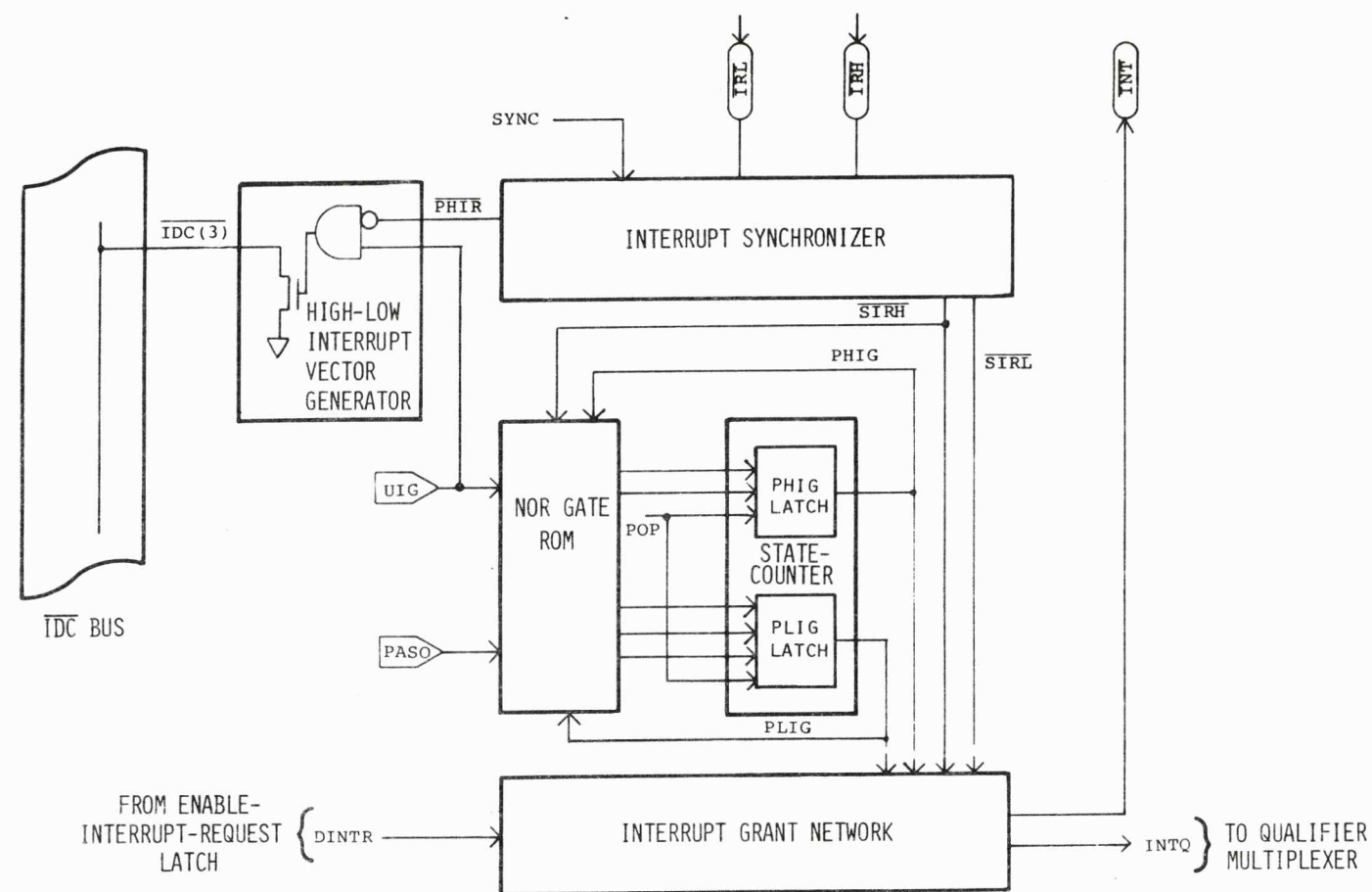


FIG 19-1-1

SECTION 19

Figure 19-1-1 is an overview of the Interrupt Controller. The purpose of the Interrupt Controller is to manage the IOC's response to the two levels of interrupt requests available to peripherals. These two levels are Interrupt Request Low (IRL) and Interrupt Request High (IRH). The Interrupt Controller uses the interrupt requests in conjunction with SYNC, the state of the Enable Interrupt Request Latch and the general state of internal affairs of the IOC, to decide whether or not to allow an interrupt. If an interrupt is to be allowed the Interrupt Controller generates $\overline{\text{INT}}$ and an internal qualifier (INTQ). $\overline{\text{INT}}$ is used by the BPC to abort the current instruction fetch and do a JSM $10_8, I$ instead. The internal qualifier INTQ is used by the IOC to initiate certain necessary internal activities to manage and respond to the interrupt.

If considered in isolation, the Interrupt Controller is not a

logically complete mechanism. That is, it both stimulates and responds to other mechanisms, both within and without the IOC. In particular, the Interrupt Controller works in conjunction with flow charting in both the Instruction Controller and Bus Controller state-machines.

The purpose of the Interrupt Synchronizer is to capture the state of the interrupt request lines upon the leading edge of SYNC. One signal is generated for each level of interrupt request. These signals are Synchronized Interrupt Request High (SIRH) and Synchronized Interrupt Request Low (SIRL).

The logic that determines whether or not to grant an interrupt on request is embodied in a 2-bit State Counter and in the Interrupt Grant Network. The State Counter consists of two latches and a NOR gate ROM. The two latches combine to represent the current interrupt status. The outputs of the two

OVERVIEW OF THE INTERRUPT PROCESS

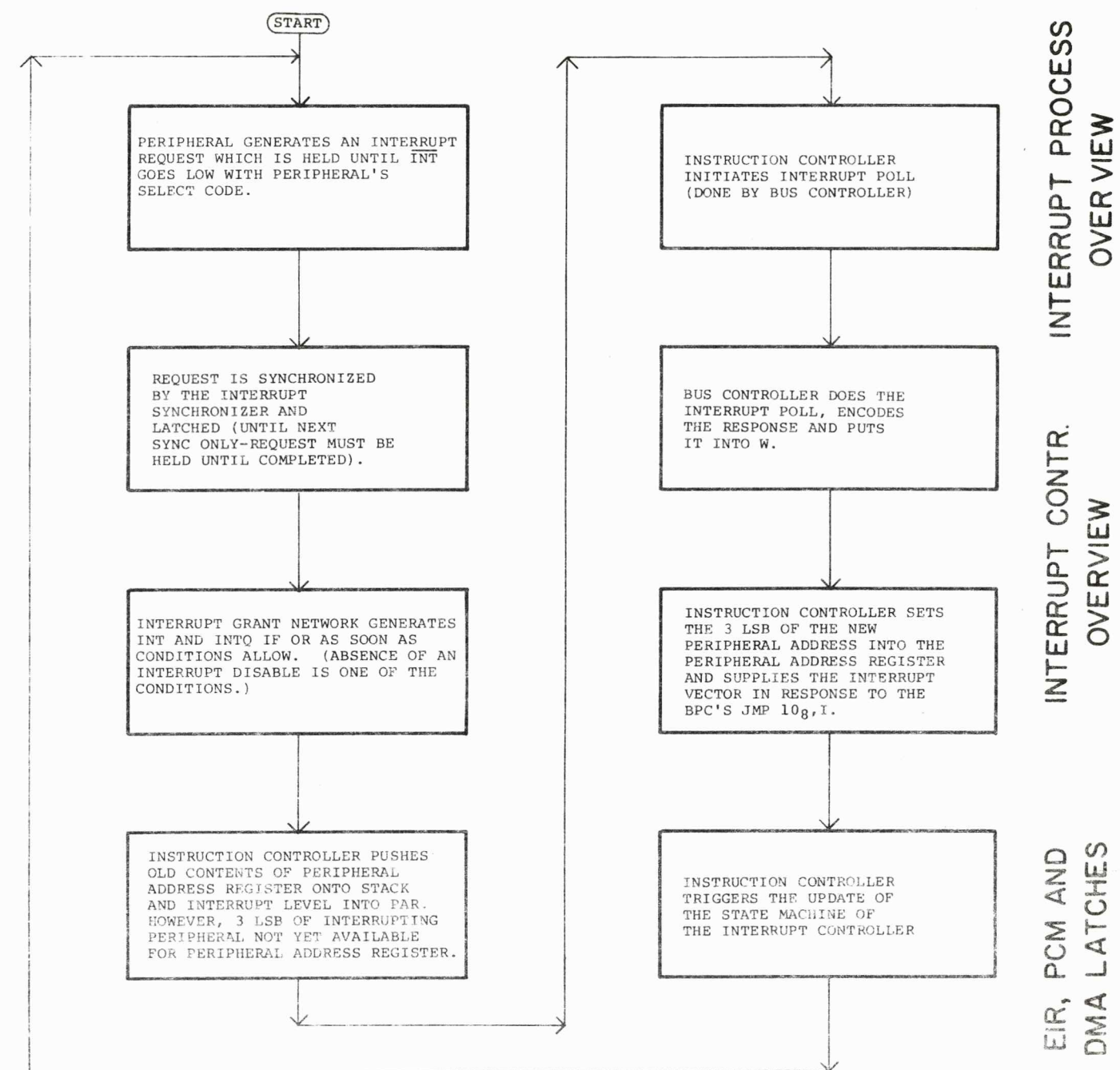


FIG 19-1-2

latches are Priority Low Interrupt Grant (PLIG) and Priority High Interrupt Grant (PHIG). The Interrupt Grant Network compares the current interrupt status against the type of request that has been made. If the request is of a higher priority than the current status, and if the interrupt system has not been disabled, the Interrupt Grant Network will generate INTQ and $\overline{\text{INT}}$.

The occurrence of $\overline{\text{INT}}$ sets the BPC's train of interrupt activity in motion. INTQ causes the Instruction Controller state machine to go

to its interrupt segment. This is the segment that forms the interrupt vector in response to the BPC's interrogation of register 10_8 . During this process the Instruction Controller issues the micro-instruction Update Interrupt Grant (UIG). UIG causes the state-machine of the Interrupt Controller to change states. The changed state will reflect the new status of the interrupt system.

The Interrupt Controller responds not only to interrupt requests but also to the events that terminate an interrupt. A success-

OVERVIEW OF THE INTERRUPT SYNCHRONIZER

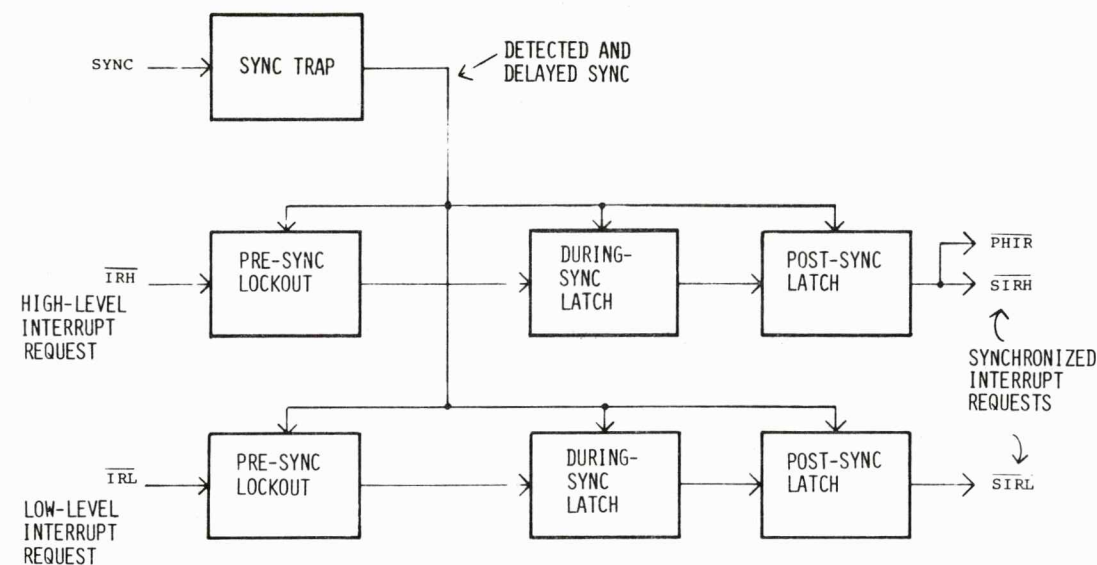


FIG 19-2-1

DETAILS OF THE INTERRUPT SYNCHRONIZER

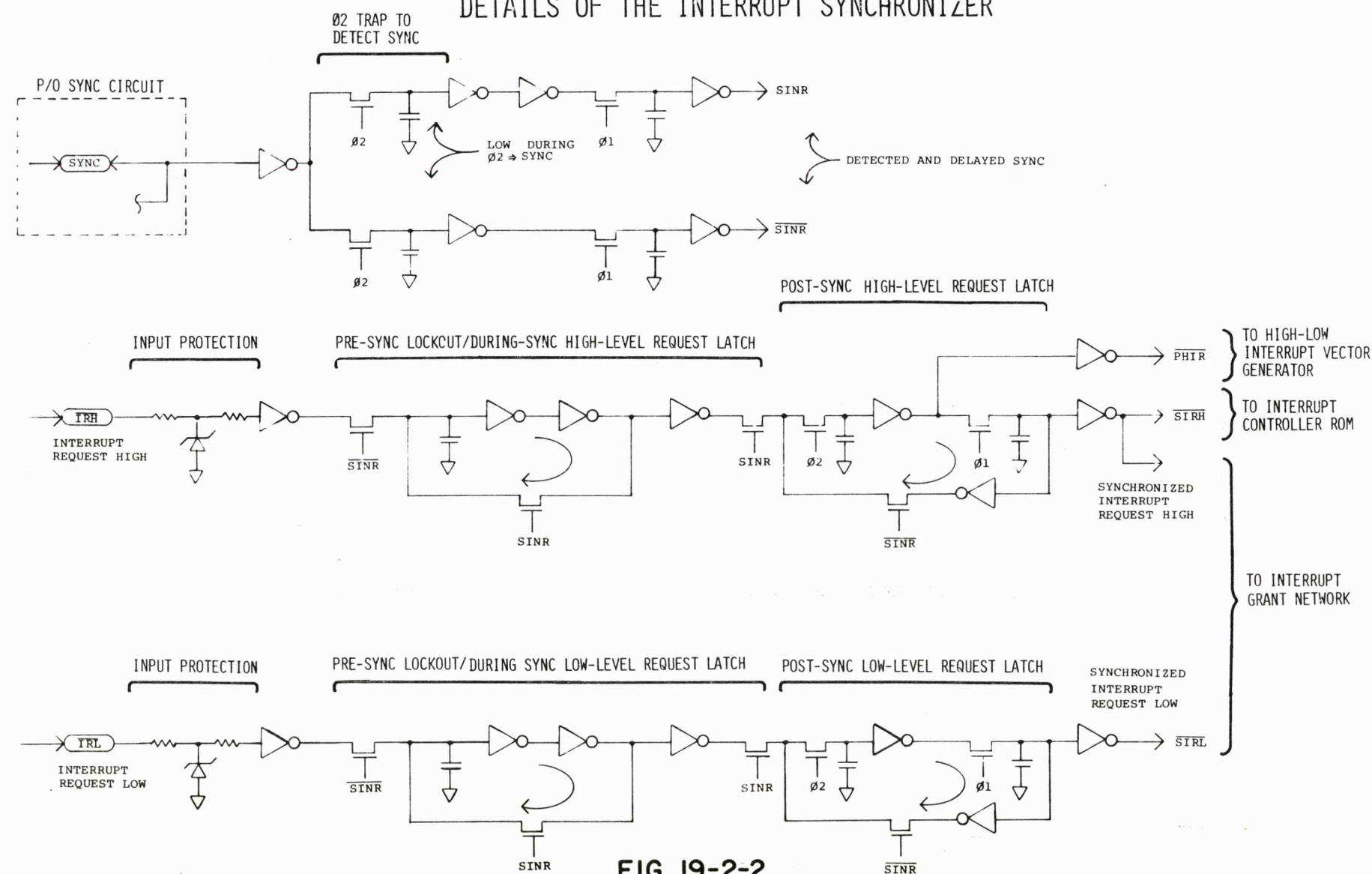


FIG 19-2-2

SECTION 19 (CONTINUED)

ful interrupt request is always accompanied by the execution of an interrupt service routine. An interrupt service routine is always concluded with the machine-instruction RET,P. The IOC is sensitive to that machine-instruction and issues the micro-instruction PASO in response to it. PASO does two things. First, it pops the Peripheral Address Stack. This concept has already been explained in conjunction with the PA register and its associated stack. Second, it informs the Interrupt Controller that the level of interrupt currently in use has been terminated. This causes a change in the State-Counter that reflects the new status of the interrupt system.

The High-Low Interrupt Vector Generator helps in forming the 4th bit of the interrupt vector, in a manner to be described later.

Figure 19-1-2 is an overview of the entire interrupt process, and is self-explanatory if you don't mind references to material that hasn't been covered yet.

Figure 19-2-1 is an overview of the Interrupt Synchronizer. The purpose of the Interrupt Synchronizer is to generate SIRL, SIRH and PHIR (identical to SIRH) in response to SYNC, IRL and IRH.

Figure 19-2-2 shows the details of the Interrupt Synchronizer. Observe how SYNC is detected and delayed to produce SINR and SINR. These two signals are used to enable the various synchronizing latches for the two levels of interrupt request.

The actual synchronization for each level of interrupt request occurs as follows. Whenever SYNC is false the current value of each level of interrupt request is loaded into a During-SYNC Latch. Those latches are closed when SYNC goes true. This constitutes the actual capture of the interrupt request.

The captured request is slightly delayed as it is fed through another latch to become either SIRH or SIRL. The purpose of these Post-SYNC Latches is to preserve the cap-

DETAILS OF THE INTERRUPT GRANT NETWORK

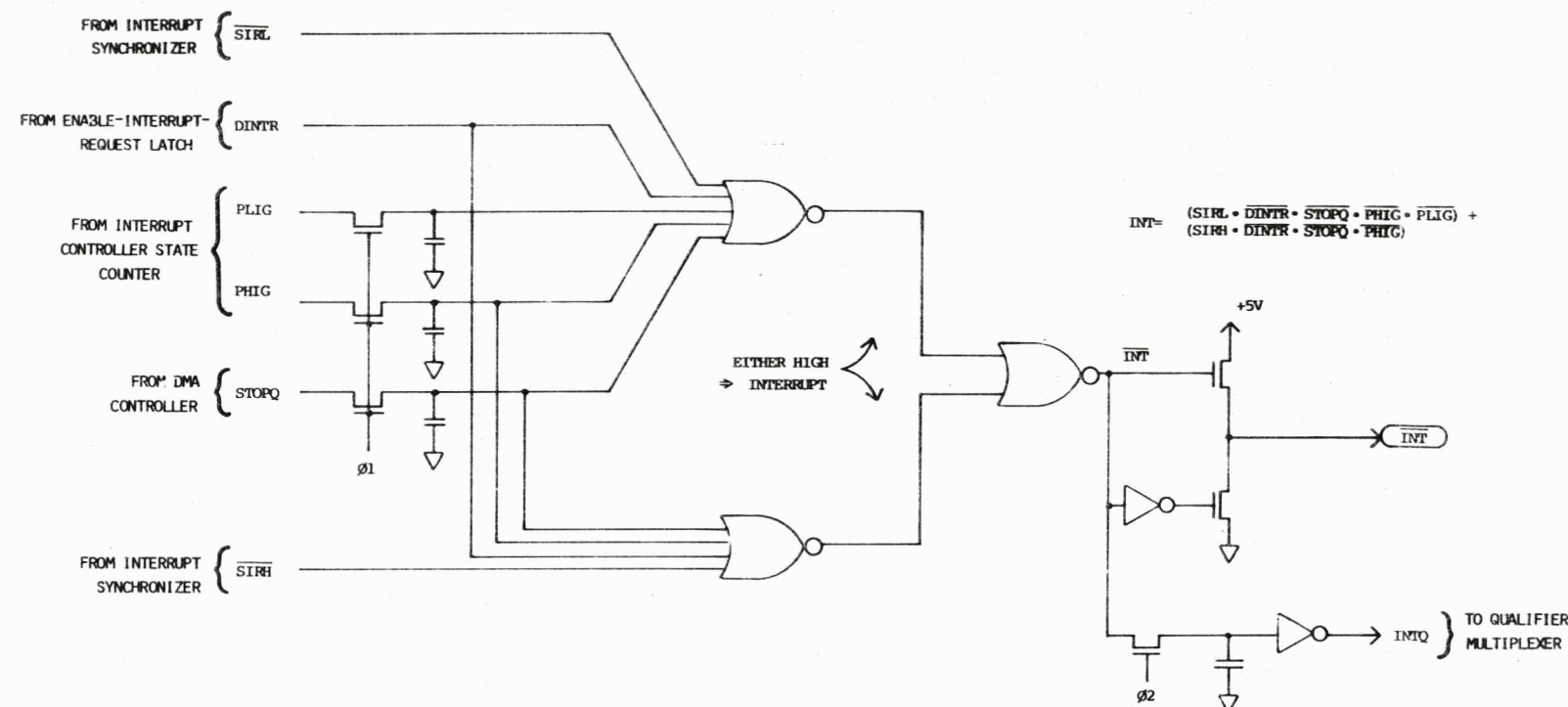


FIG 19-3-1

SECTION 19 (CONTINUED)

tured value even through SYNC may again go false. As soon as SYNC goes false the During-SYNC Latches must be available to capture a subsequent request.

Figure 19-3-1 shows the details of the Interrupt Grant Network. The purpose of this circuitry is to respond to the conditions in which an interrupt should be granted. The circuitry forms the OR of two general conditions. These conditions are as follows. First condition: a synchronized low level interrupt request, absence of a disable interrupt signal, absence of \overline{STOPQ} (meaning no Bus Request in progress), absence of high level interrupt in progress and absence of a low level interrupt in progress. Second condition: a synchronized high level interrupt request, absence of a disable interrupt system signal, absence of a \overline{STOPQ} and absence of an ongoing high level interrupt. If either of these two general conditions is met, \overline{INTQ} is generated and \overline{INT} is pulled low.

Figure 19-3-2 is a flow chart illustrating the relationship between the Interrupt Grant Network and other interrupt related activities. In essence, it summarizes the possible situations surrounding the activity of the Interrupt Grant Network. Roughly speaking, the drawing is self-explanatory.

RELATIONSHIP BETWEEN THE INTERRUPT GRANT NETWORK AND OTHER INTERRUPT RELATED ACTIVITY

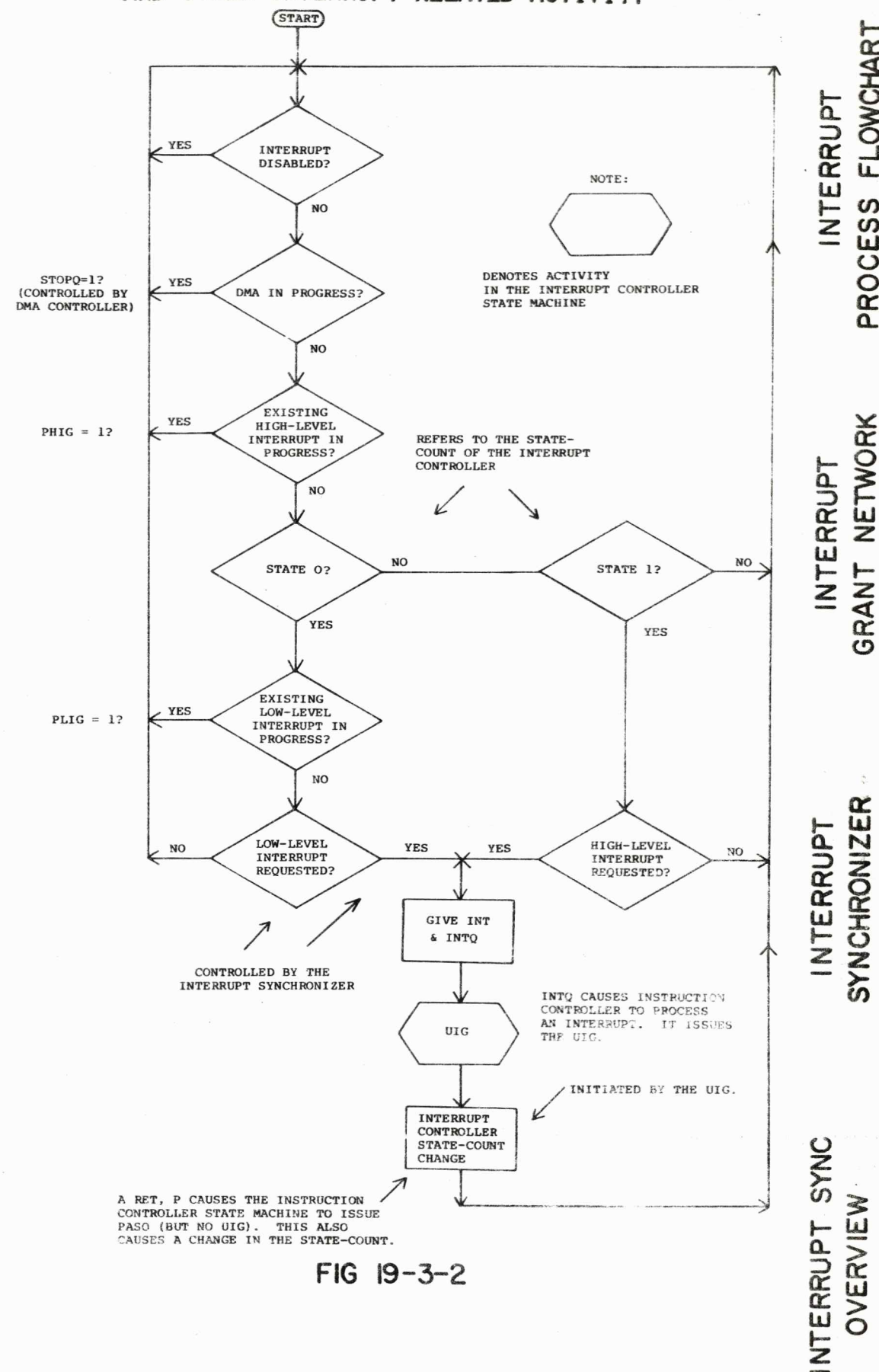


FIG 19-3-2

DETAILS OF THE ROM AND STATE COUNTER OF THE INTERRUPT CONTROLLER

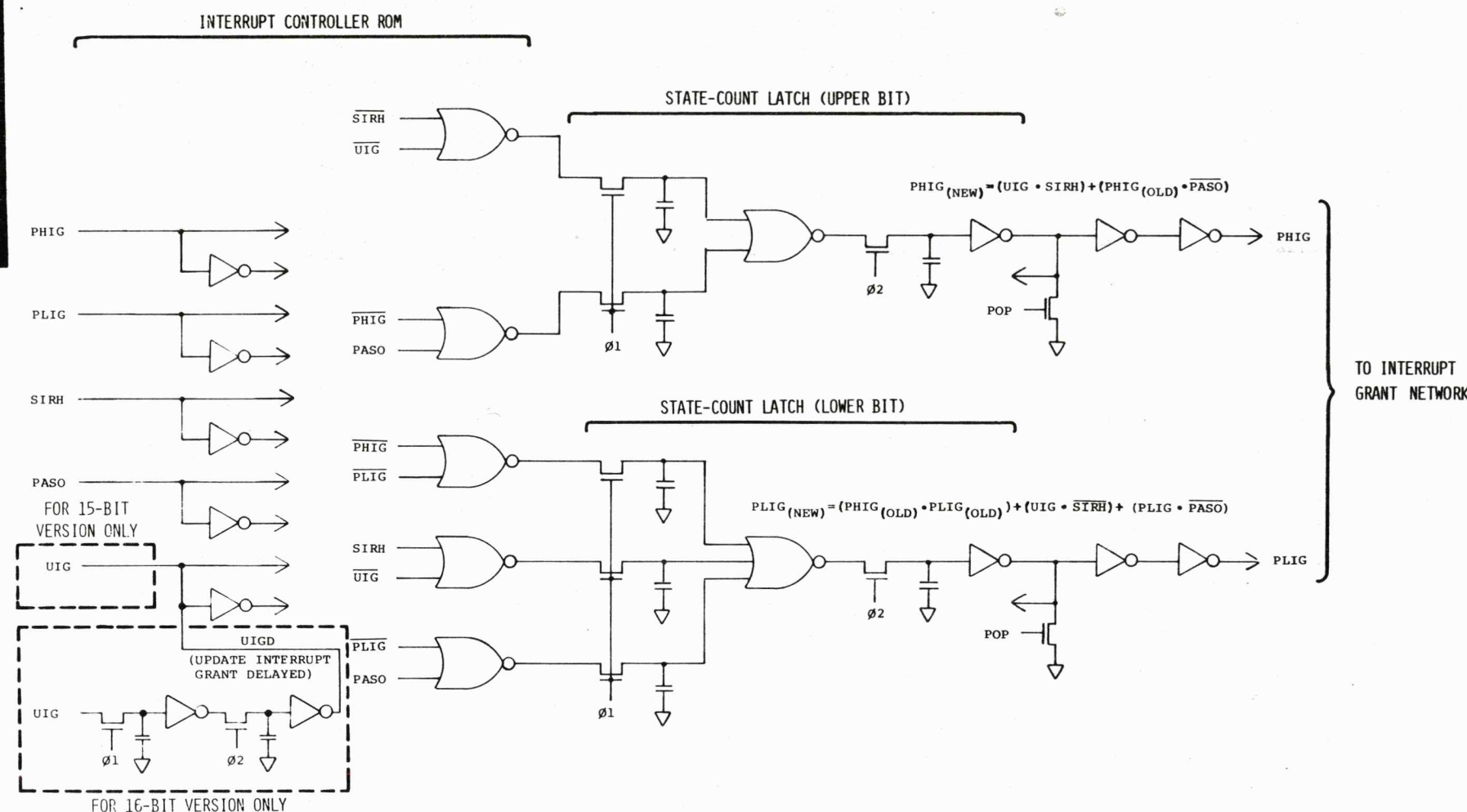
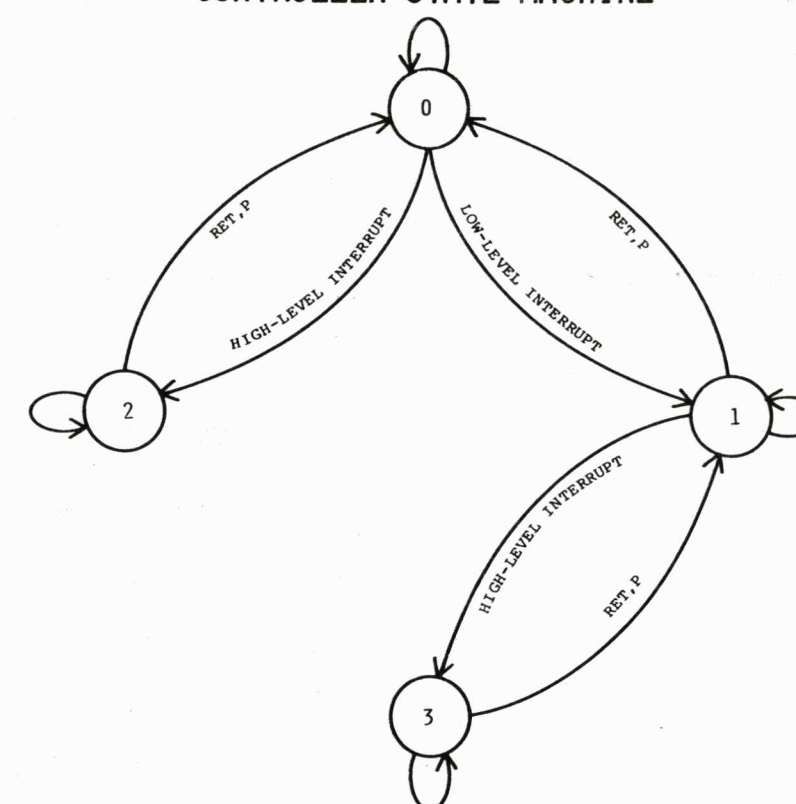


FIG 19-4

OVERVIEW OF THE INTERRUPT CONTROLLER STATE MACHINE



INTERRUPT CONTROLLER STATE ASSIGNMENTS

- 0 NO INTERRUPTS IN PROGRESS
- 1 SOLITARY LOW-LEVEL INTERRUPT IN PROGRESS
- 2 SOLITARY HIGH-LEVEL INTERRUPT IN PROGRESS
- 3 LOW-LEVEL HAS BEEN INTERRUPTED BY HIGH-LEVEL

FIG 19-5-1

SECTION 19 (CONTINUED)

Figure 19-4 illustrates the details of the NOR gate ROM and of the 2-bit State Counter for the Interrupt Controller. The exact meanings of the possible state counts, and of the logic behind their transitions, is explained in subsequent drawings.

There is a minor difference between the 15 and 16-bit versions concerning Update Interrupt Grant (UIG). What is involved here is the bug in the 15-bit version concerning the number of levels of indirect addressing associated with the interrupt vector. This bug is described in the N-MOS II Processor Manual. It has to do with the IOC's releasing INT at exactly the wrong time. This allows a race condition to exist in the event that only a single level of indirect is called

for by the interrupt vector. The solution was to alter the IOC so that it holds INT down longer. This was done by inserting a delay in series with UIG. Thus, in the 16-bit version PLIG and PHIG change one state later than they do in the 15-bit version. Referring back to the Interrupt Grant Network, it will be seen that this delayed change in state causes INT to be held one state longer.

Figure 19-5-1 is a state diagram of the Interrupt Controller's state assignments. The state numbers represent the binary number obtained by designating PHIG as the most significant bit and PLIG as the least significant bit, respectively, of the state-count. When no interrupts are in progress both PLIG and PHIG are false. Thus, state 0 can be thought

of as the idling state of the Interrupt Controller state machine. Two transitions are possible out of state 0. A high level interrupt request will cause a transition to state 2. Two is the binary count obtained when PHIG is true PLIG is false. The PASO associated with the RET,P will cause state 2 to transition back to state 0. It is worth noting that state 2 corresponds to the occurrence of a high level interrupt with no previous low level interrupt in progress.

The other transition from state 0 is to state 1. This occurs when a low level interrupt is requested with no previous interrupt in progress. State 1 is designated by PHIG false and PLIG true. State 1 can transition back to state 0 as a result of a RET,P, or to state

3 if a high level interrupt request should occur. State 3 differs from state 2 in that when the high level interrupt is completed the previous low level interrupt activity is resumed, by transitioning back to state 1 as opposed to state 0. State 3 is designated by both PHIG and PLIG being true. This is a fine notational device, since in the case of state 3 both interrupt levels have been invoked.

Figures 19-5-2 through 19-5-5 illustrate the various possible state count transitions of the Interrupt Controller state-machine. A logical description of each state is shown in all its gory detail.

DETAILS OF STATE 0 OF THE INTERRUPT CONTROLLER STATE MACHINE

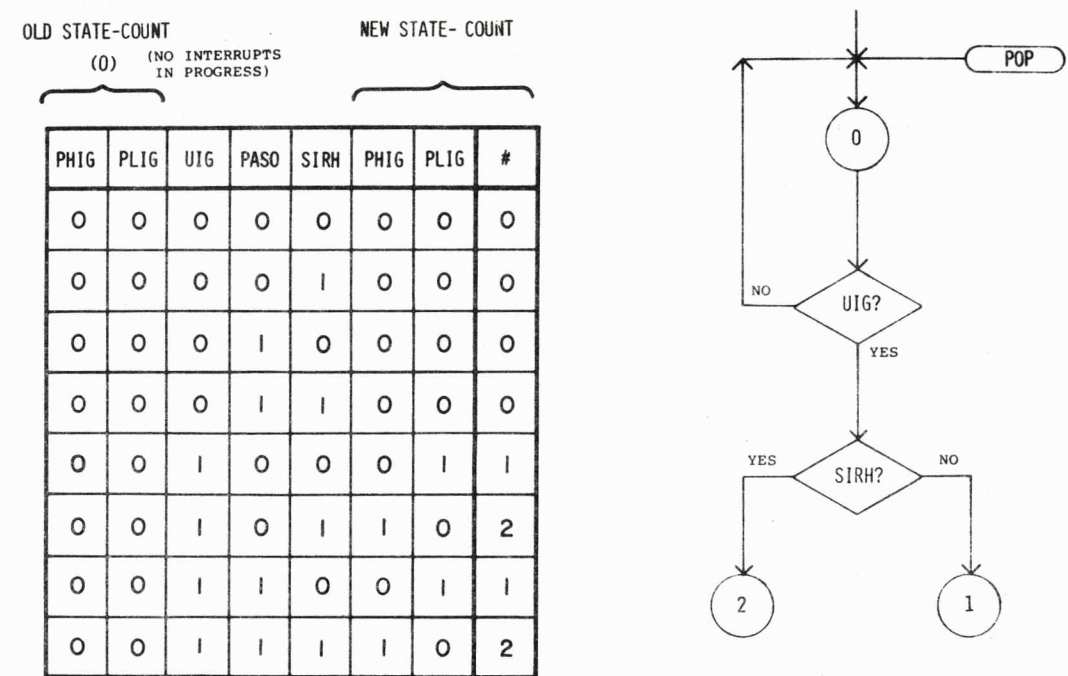
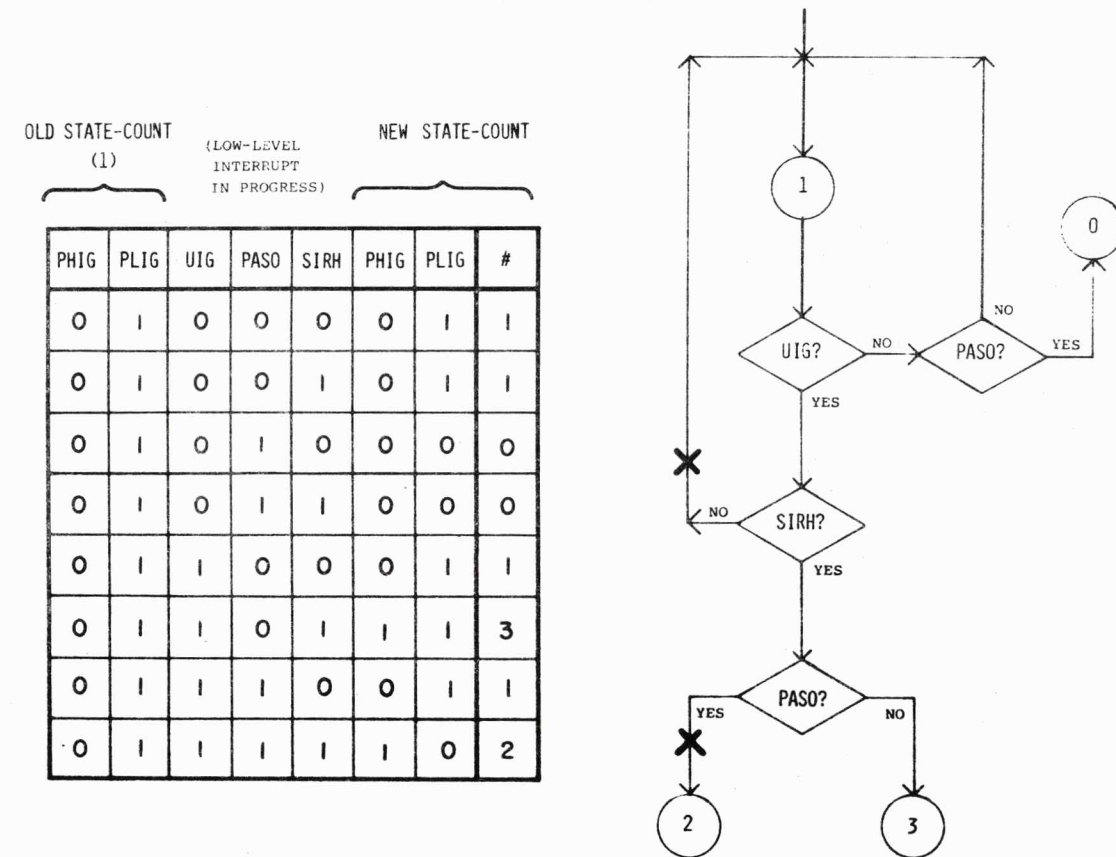


FIG 19-5-2

DETAILS OF STATE 1 OF THE INTERRUPT CONTROLLER STATE MACHINE



X DENOTES A TRANSITION THAT DOES NOT OCCUR

FIG 19-5-3

DETAILS OF STATE 2 OF THE INTERRUPT CONTROLLER STATE MACHINE

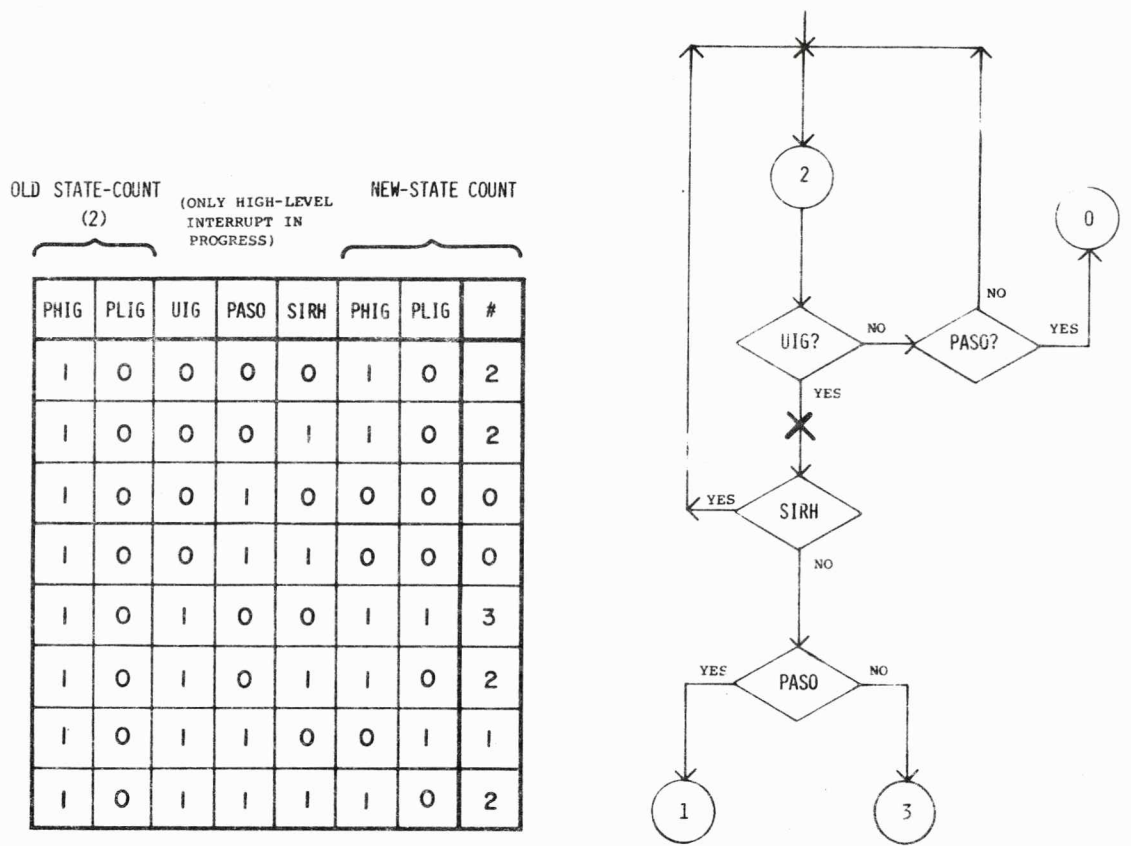
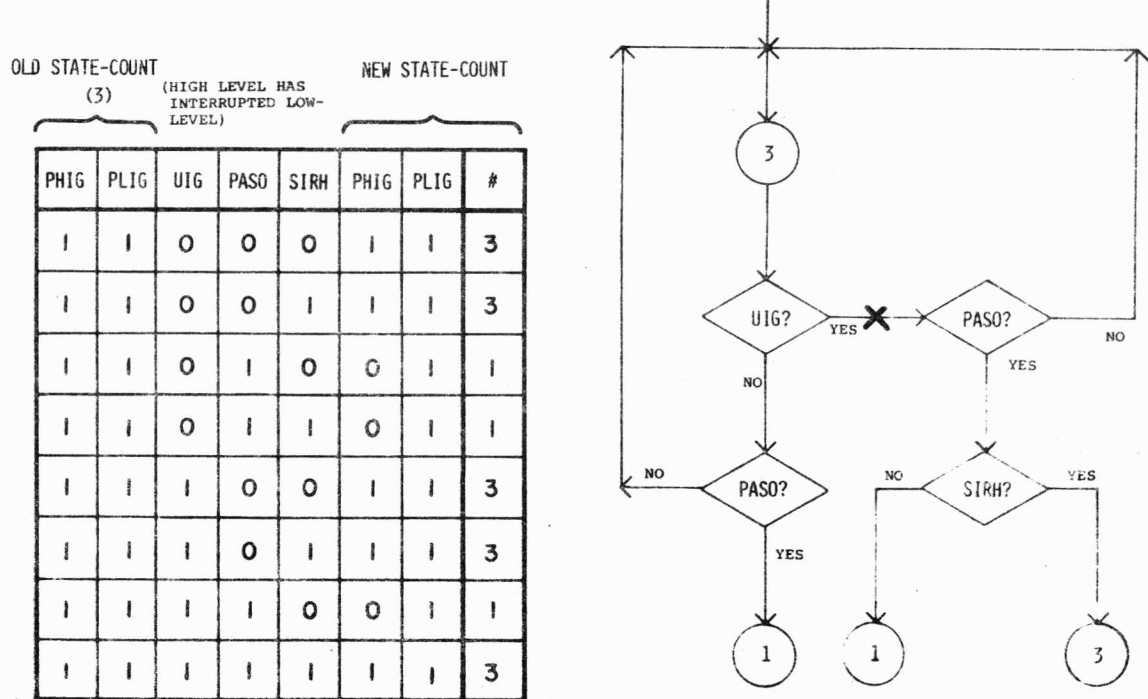


FIG 19-5-4

X DENOTES A TRANSITION THAT DOES NOT OCCUR

DETAILS OF STATE 3 OF THE INTERRUPT CONTROLLER STATE MACHINE



X DENOTES A TRANSITION THAT DOES NOT OCCUR

FIG 19-5-5

INTR. CONTR. STATE 2

INTR. CONTR. STATE 0

INTR. CONTR. STATE 1

INTR. CONTR. STATE MACHINE OVERVIEW

106

FLOW CHART OF THE INTERRUPT CONTROLLER STATE MACHINE

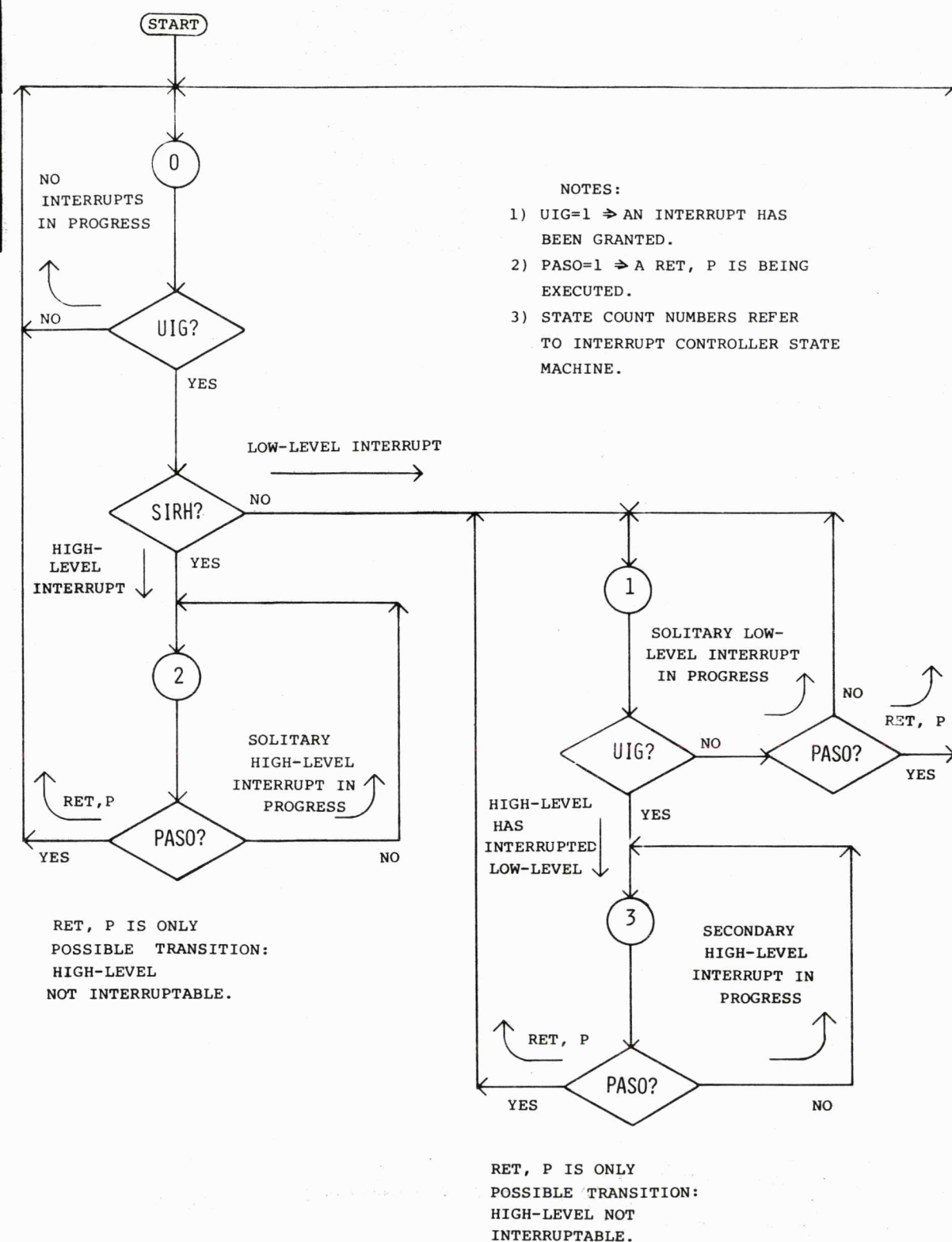


FIG 19-5-6

DETAILS OF THE HIGH-LOW INTERRUPT VECTOR GENERATOR

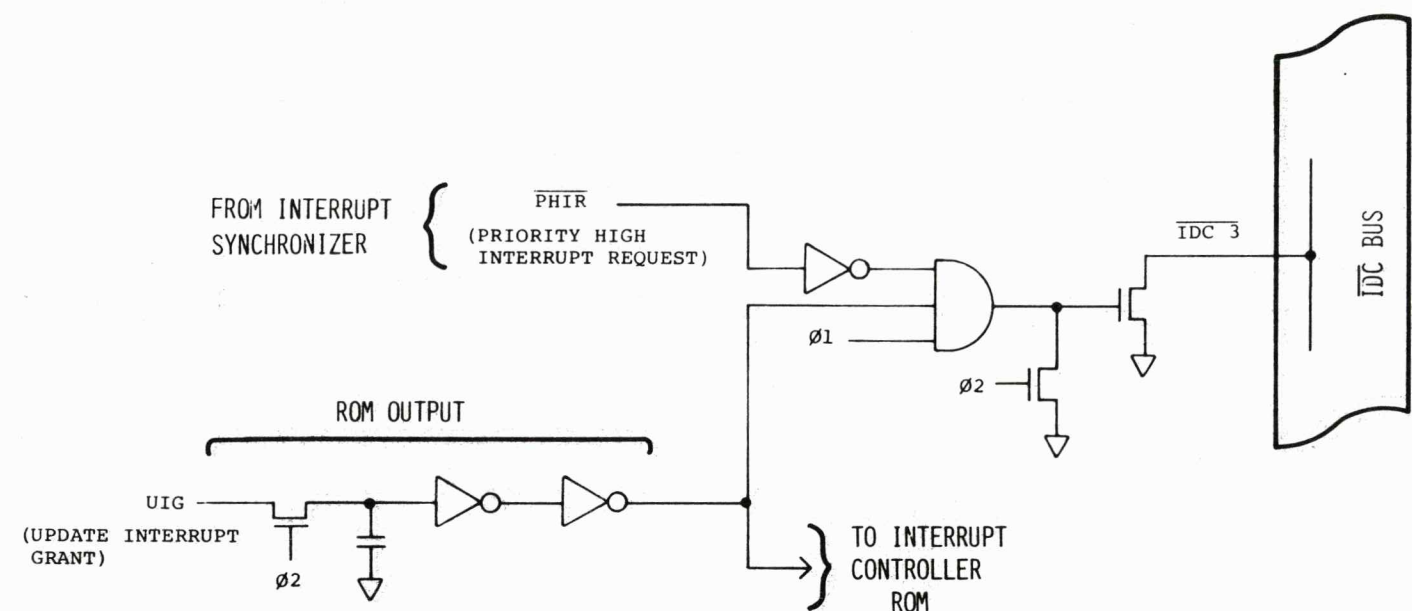


FIG 19-6

SECTION 19 (CONTINUED)

Figure 19-5-6 is a compilation of the individual flow chart segments into a useable state-machine diagram. This document should be taken as the primary description of the Interrupt Controller state-machine.

Figure 19-6 shows the details of the High-Low Interrupt Vector Generator. The purpose of this circuitry is to help form 4th bit of the interrupt vector. Recall that the bottom three bits come from the Select Code Priority Resolver associated with the W register. Recall also that the upper 12 bits come from the IV register. The 4th bit corresponds to the level of interrupt that is in progress. That is, either a high level request or a low level request. If it's not the high level request then it must be the low level request. The PHIR on the drawing is the same as SIRH from the Interrupt Synchronizer. That is, PHIR corresponds uniquely to either a high or low level of interrupt request. Now, it turns out that the only time UIG is given is when the interrupt vector is being formed on the IDC Bus. So, besides servicing the Interrupt Controller ROM, UIG causes PHIR to be placed on the IDC Bus at the bit 3 position.

OVERVIEW OF THE DMA CONTROLLER

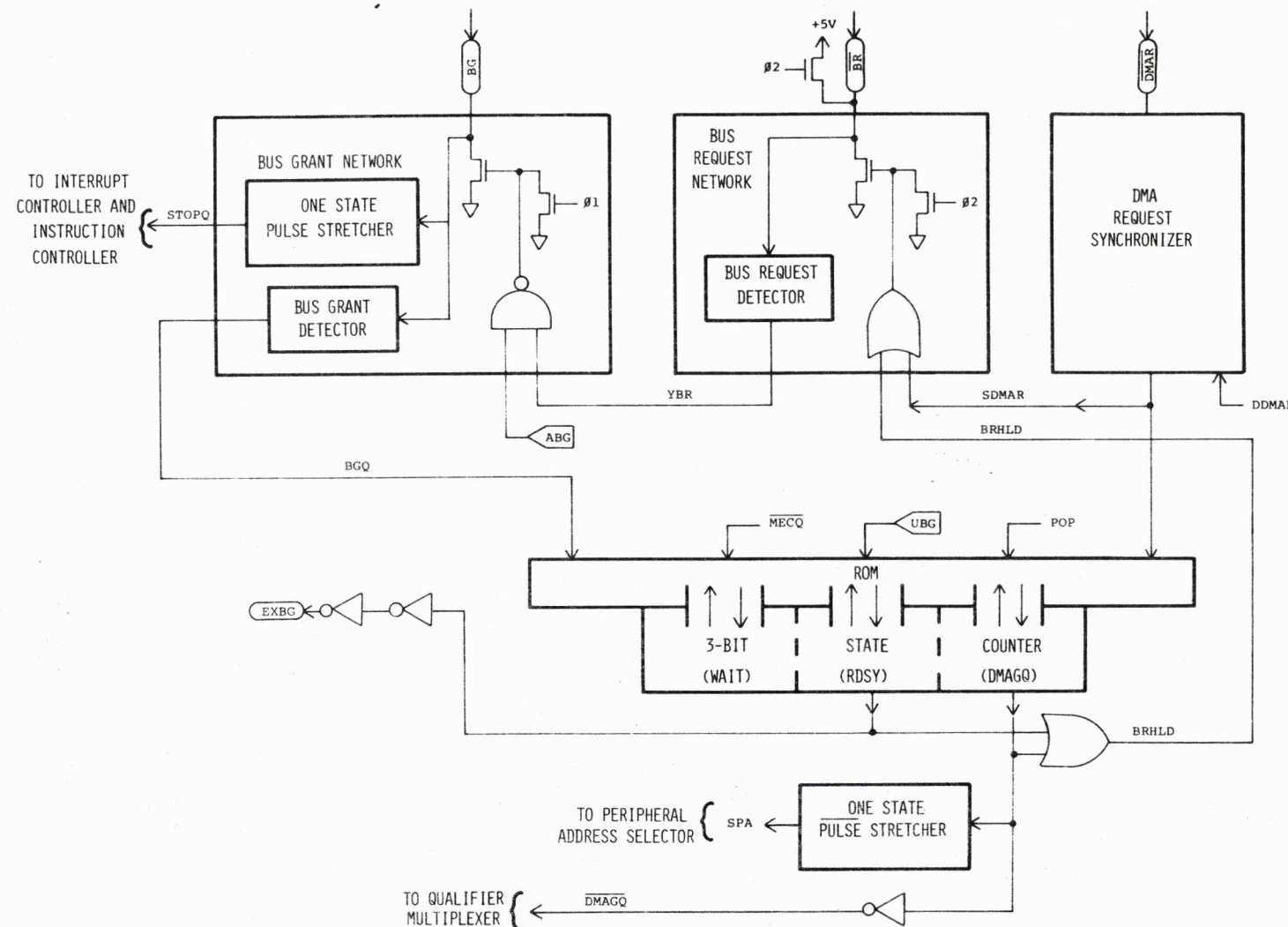


FIG 20-1-1

SECTION 20

This section deals with the DMA Controller. Like the Interrupt Controller the DMA Controller consists of a small state-machine assisted by some dedicated circuitry. Also as in the case of interrupt, substantial portions of DMA related activity occur elsewhere than in the DMA Controller. In this case, the Bus Controller has substantial sections of flow charting that deal with DMA and the Pulse Count Mode. Considered as a whole, the DMA process can only be understood as the interaction between the DMA Controller and the Bus Controller.

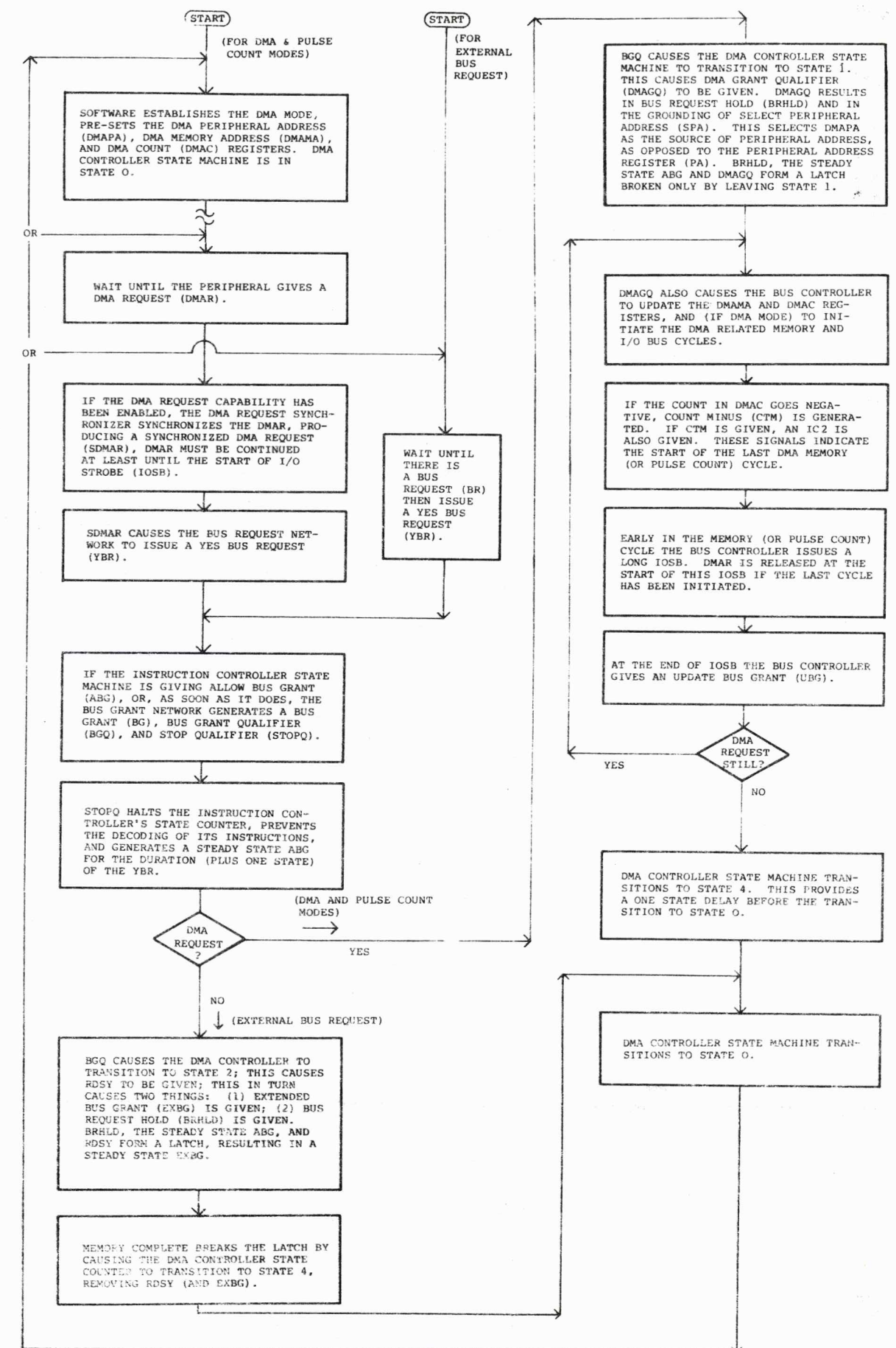
Obviously then, this section will not explain the entire DMA process as a unit. The flow charting implemented by the Bus Controller is covered in another section. Our approach here will be to say a few

words over each box in the overview of the DMA Controller shown in Figure 20-1-1. We will rely on the summary shown in Figure 20-1-2 to relate the sequential actions of each block in the overview and to describe their general relationship to the activities of the Bus Controller.

Figure 20-1-1 is an overview of the DMA Controller. The purpose of the DMA Controller is to respond to requests for DMA or simple bus requests by controlling Bus Grant and Extended Bus Grant (EXBG), while generating qualifier information for the Bus Controller.

The DMA Request Synchronizer allows asynchronous grounding of DMA Request (DMAR). It also incorporates the function of disabling the DMA Request line. A successful DMAR

OVERVIEW OF THE DMA, PULSE COUNT AND BUS REQUEST PROCESSES



NOTE: BUS GRANT (BG) IS GIVEN FOR AS LONG AS THE BRHLD LATCH IS IN EFFECT.

FIG 20-1-2

DMA, PULSE COUNT
AND BUS REQUEST
PROCESSES OVERVIEW

DMA CONTROLLER
OVERVIEW

HIGH-LOW
INTERRUPT
VECTOR GENERATOR

INTERRUPT
CONTROLLER
STATE MACHINE

DETAILS OF THE DMA REQUEST SYNCHRONIZER

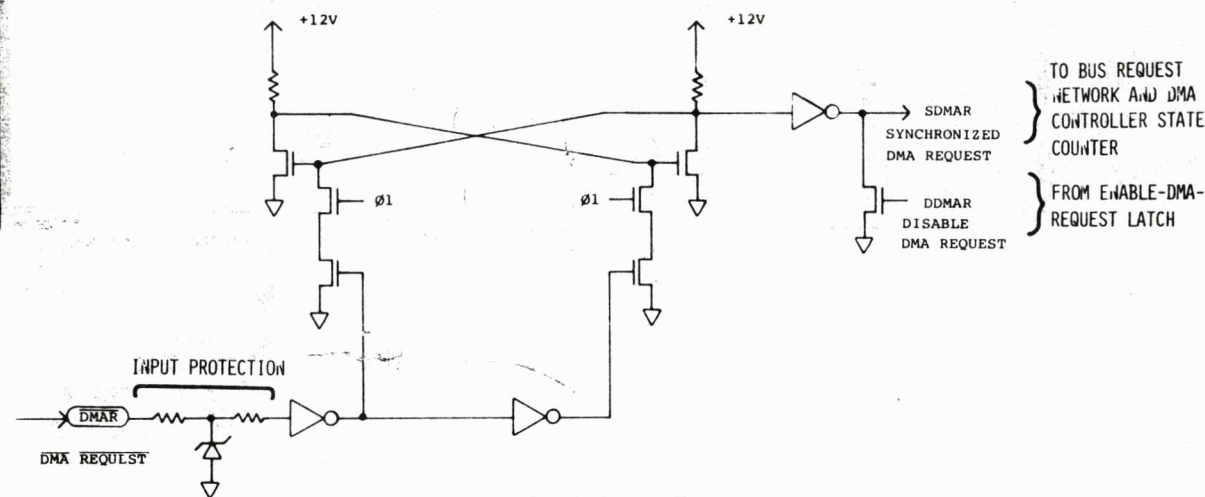


FIG 20-2

SECTION 20 (CONTINUED)

produces a Synchronized DMA Request (SDMAR) which is stable during phase two. SDMAR is sent to the Bus Request Network, where it is used to generate a Bus Request.

The Bus Request Network is the IOC's means to generate a Bus Request. The Bus Request Network will ground \overline{BR} when either of two signals are true. There are SDMAR and Bus Request Hold (BRHLD). SDMAR must generate a Bus Request, since the Bus Request/Bus Grant mechanism is the basis for DMA operation. Bus Request Hold is a signal that is generated whenever a Bus Grant is in progress, and allows the peripheral requesting activity to release \overline{DMAR} or \overline{BR} . Bus Request Hold will keep \overline{BR} grounded until the DMA operation or simple Bus Request is "concluded" (indicated by Memory Complete). This is basically a service function for the peripheral, to make the use of \overline{DMAR} and \overline{BR} easier.

A second function of the Bus Request Network is to detect the fact that any agency has performed a Bus Request. It represents that fact with the signal YBR (Yes Bus Request).

The Bus Grant Network keeps BG false unless two conditions are met. These conditions are first, that there is an actual Bus Request (YBR is true), and second, that the Instruction Controller indicates

with ABG that it is permissible to allow a Bus Grant.

Another function of the Bus Grant Network is to detect an actual Bus Grant. It uses that to generate STOPQ and BGQ (Bus Grant Qualifier).

The DMA Controller State-Machine has two main purposes. First, it represents the current bus request status. These are: no bus request in progress; a simple non-DMA related bus request in progress; and finally, DMA request in progress. The second major function of the DMA Controller State-Machine is to serve as an interlock for changing from one bus request status to another. (The actual activity corresponding to a given status is controlled elsewhere.) DMA requests are forwarded as DMAGQ and SPA, as part of assuming the status of DMA-in-progress. Likewise, an Extended Bus Grant is generated whenever a simple bus request, by an agency further down the daisy chain from the IOC, is granted. As part of the second major function the State-Machine also responds to various completion indicators. These are MEC and Update Bus Grant (UBG).

The organization of the state-machine is relatively simple. It has a 3-bit State-Counter which is set to all zero's at turn-on by POP. The all zero condition indicates that no bus request activity of any kind

DETAILS OF THE BUS REQUEST NETWORK

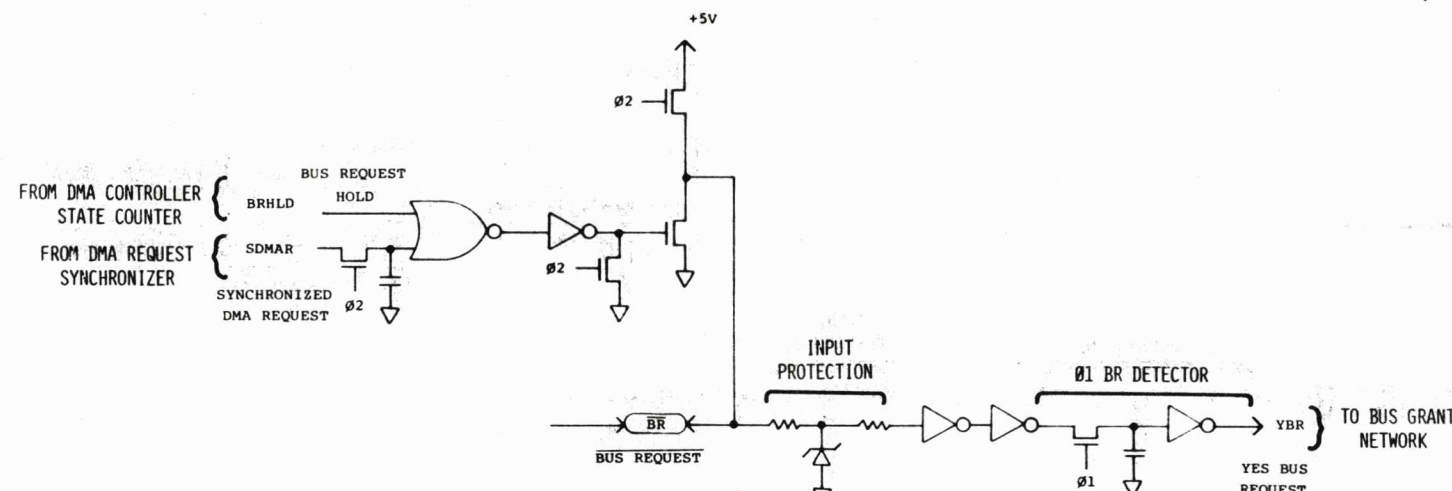


FIG 20-3

is in progress. For purposes of assigning binary state counts the 3 bits are ordered in the following way. The least significant bit represents the DMA condition and generates DMAGQ. The middle bit represents the occurrence of a non-DMA external Bus Request. This bit generates Extended Bus Grant. The most significant bit creates a transitional state used as a delay between the end of an activity and the return to the idle condition. Only one bit of the State Counter will be set at a time.

Both the DMA and Pulse Count Modes work the same as far as the DMA Controller is concerned. The difference between these two types of activity is found in the flow charting of the Bus Controller.

Figure 20-1-2 is an overview of the DMA, Pulse Count and Bus Request processes. This is the best summary of how a peripheral and the elements within the IOC interact during those various processes. Roughly speaking, the document is self-explanatory.

Figure 20-2 shows the details of the DMA Request Synchronizer. Basically, it is a cross-coupled flip/flop which is immune to its inputs except during phase one. Observe how SDMAR is made false if DDMA is true.

Figure 20-3 shows the details of the Bus Request Network. Observe how either of BRHLD or SDMAR can cause a Bus Request. \overline{BR} is a pre-charged signal and can be grounded only during phase one. To ensure proper handling of \overline{BR} the OR'ing of BRHLD and SDMAR must be done during phase two. BRHLD is itself the OR of two other signals, both of which change at the leading edge of phase two. SDMAR however, changes during phase one. Hence, SDMAR is routed through an extra phase two transfer gate before being OR'ed with BRHLD.

The remaining circuitry constitutes a Bus Request Detector. Bus Request is truthful during phase one. The Bus Request Detector is simply a phase one transfer gate which looks at Bus Request only during phase one.

DETAILS OF THE BUS GRANT NETWORK

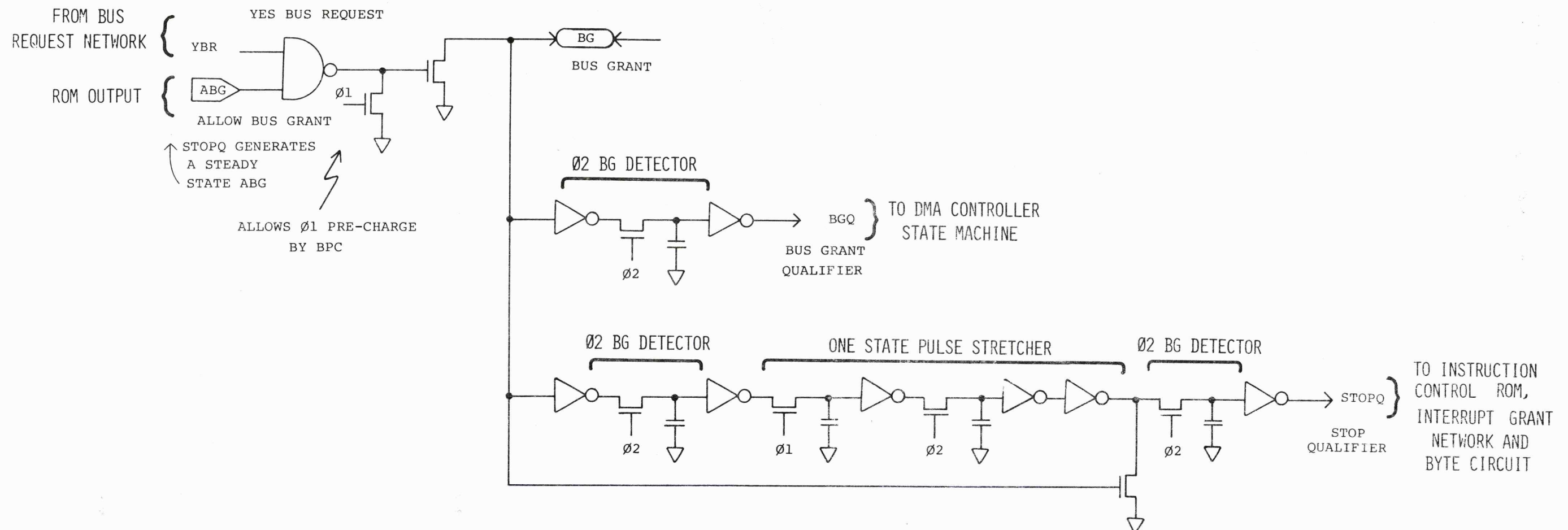


FIG 20-4

SECTION 20 (CONTINUED)

Figure 20-4 shows the details of the Bus Grant Network. On the one hand, the Bus Grant Network allows Bus Grant to go true (it is given by the BPC), provided that an actual request has been made, and that the Instruction Controller indicates (via ABG) that a Bus Grant is permissible. This does *not* mean that Bus Grant will actually go true. Some other agency further down the IDA Bus may decline to allow Bus Grant to go true by keeping it grounded out.

On the other hand, a successful Bus Grant is detected by two phase two Bus Grant Detectors. One detector is used to generate BGQ, which signals the DMA Controller State-Machine. The other detection mechanism is employed to generate STOPQ. A one-state pulse stretcher

driven by yet another detector ensures that STOPQ will last one state longer than Bus Grant.



FIG 20-5



DETAILS OF EXBG, BRHLD, $\overline{\text{DMAGQ}}$, AND SPA

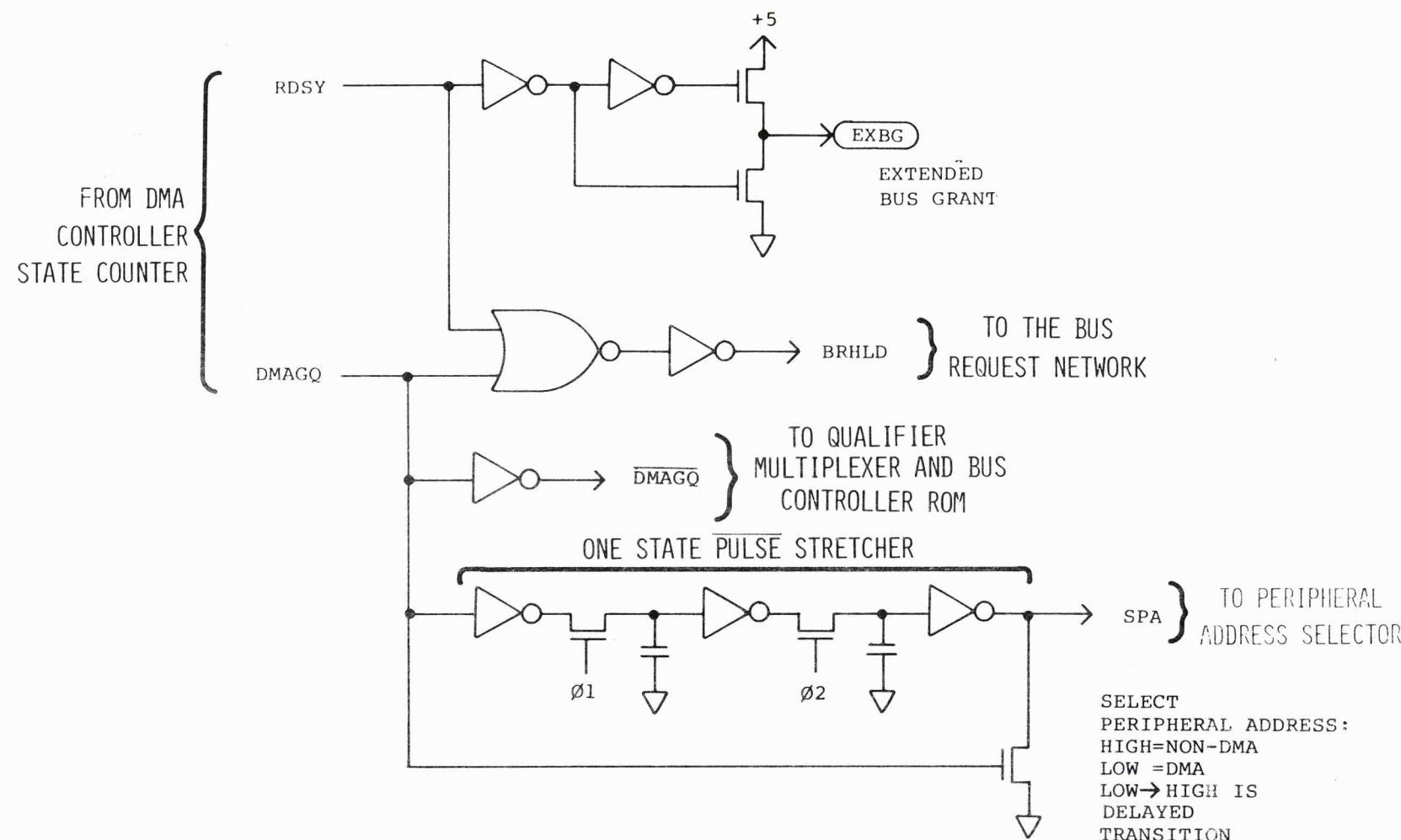
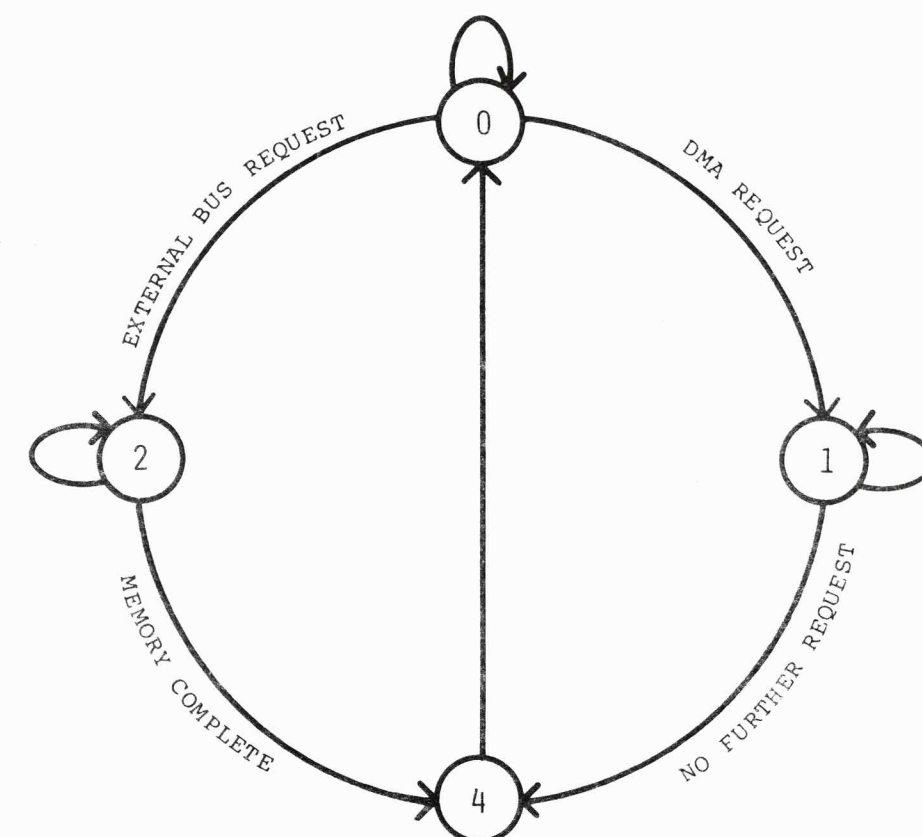


FIG 20-6

OVERVIEW OF THE DMA CONTROLLER STATE MACHINE



DMA CONTROLLER STATE ASSIGNMENTS

- ① NO DMA OR BUS REQUESTS IN PROGRESS
- ② DMA REQUEST HAS BEEN GRANTED
- ③ EXTERNAL BUS REQUEST HAS BEEN GRANTED
- ④ ONE STATE DELAY

FIG 20-7-1

SECTION 20(CONTINUED)

Figure 20-6 illustrates the generation of EXBG, BRHLD, $\overline{\text{DMAGQ}}$ and SPA. EXBG is logically identical with the output RDSY from the DMA Controller State-Counter. BRHLD is simply the OR of the two state-count latches that represent ongoing bus request activity. That is, BRHLD is given whenever there is DMA activity in progress, or whenever there is an Extended Bus Grant. SPA is essentially a stretched $\overline{\text{DMAGQ}}$, and is used to determine which of the PA and DMAPA registers controls the Peripheral Address Bus.

Figure 20-7-1 illustrates the basic scheme of the DMA Controller State-Machine. State four is used as the delay when returning to state zero from either of states one or two.

Figures 20-7-2 through 20-7-5 illustrate in all their gory detail the complete breakdown of each state in the DMA Controller State-Machine.

DETAILS OF THE DMA CONTROLLER STATE MACHINE

OLD STATE COUNT (0)							NEW STATE COUNT			
WAIT	RDSY	DMAGQ	MECQ	UBG	BGQ	SDMAR	WAIT	RDSY	DMAGQ	#
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	1	0	2
0	0	0	0	0	1	1	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0
0	0	0	0	1	1	0	0	1	0	2
0	0	0	0	1	1	1	0	0	1	1
0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0	2
0	0	0	1	0	1	1	0	0	1	1
0	0	0	1	1	0	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0	0
0	0	0	1	1	1	0	0	1	0	2
0	0	0	1	1	1	1	0	0	1	1

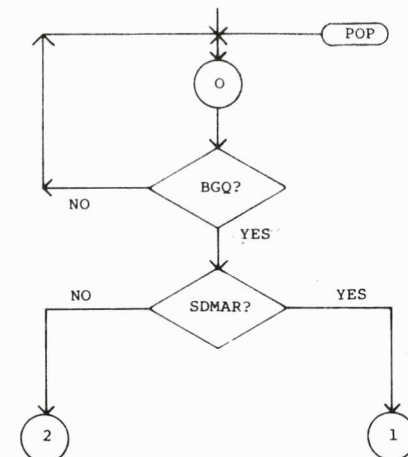


FIG 20-7-2

DETAILS OF THE DMA CONTROLLER STATE MACHINE

OLD STATE COUNT (2)							NEW STATE COUNT			
WAIT	RDSY	DMAGQ	MECQ	UBG	BGQ	SDMAR	WAIT	RDSY	DMAGQ	#
0	1	0	0	0	0	0	0	1	0	2
0	1	0	0	0	0	1	0	1	0	2
0	1	0	0	0	1	0	0	1	0	2
0	1	0	0	0	1	1	0	1	0	2
0	1	0	0	1	0	0	0	1	0	2
0	1	0	0	1	0	1	0	1	0	2
0	1	0	0	1	1	0	0	1	0	2
0	1	0	0	1	1	1	0	1	0	2
0	1	0	1	0	0	0	1	0	0	4
0	1	0	1	0	0	1	1	0	0	4
0	1	0	1	0	1	0	1	0	0	4
0	1	0	1	0	1	1	1	0	0	4
0	1	0	1	1	0	0	1	0	0	4
0	1	0	1	1	0	1	1	0	0	4
0	1	0	1	1	1	0	1	0	0	4
0	1	0	1	1	1	1	1	0	0	4

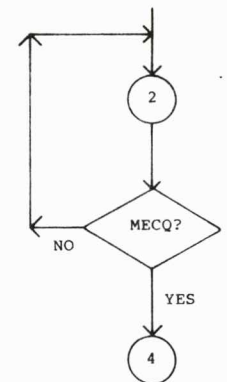
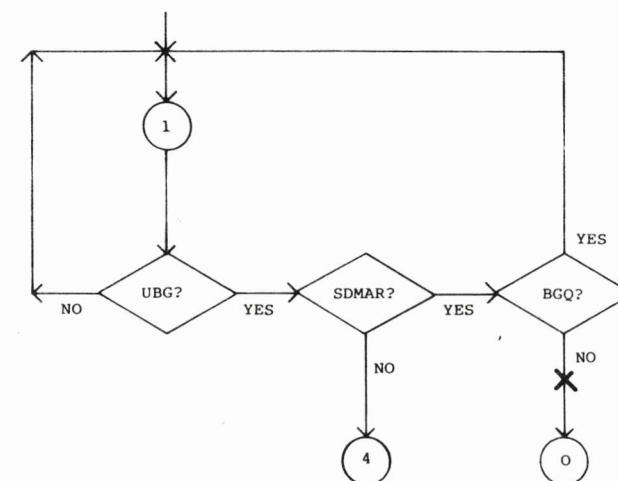


FIG 20-7-4

DETAILS OF THE DMA CONTROLLER STATE MACHINE

OLD STATE COUNT (1)							NEW STATE COUNT			
WAIT	RDSY	DMAGQ	MECQ	UBG	BGQ	SDMAR	WAIT	RDSY	DMAGQ	#
0	0	1	0	0	0	0	0	0	1	1
0	0	1	0	0	0	1	0	0	1	1
0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	1	1	0	0	1	1
0	0	1	0	1	0	0	1	0	0	4
0	0	1	0	1	0	1	0	0	0	0
0	0	1	0	1	1	0	1	0	0	4
0	0	1	0	1	1	1	0	0	1	1
0	0	1	1	0	0	0	0	0	1	1
0	0	1	1	0	0	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	1
0	0	1	1	0	1	1	0	0	1	1
0	0	1	1	1	0	0	1	0	0	4
0	0	1	1	1	0	1	0	0	0	0
0	0	1	1	1	1	0	1	0	0	4
0	0	1	1	1	1	1	0	0	1	1



X DENOTES A TRANSITION THAT DOES NOT OCCUR

FIG 20-7-3

DETAILS OF THE DMA CONTROLLER STATE MACHINE

OLD STATE COUNT (4)							NEW STATE COUNT			
WAIT	RDSY	DMAGQ	MECQ	UBG	BGQ	SDMAR	WAIT	RDSY	DMAGQ	#
1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	1	1	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0
1	0	0	0	1	1	0	0	0	0	0
1	0	0	0	1	1	1	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0
1	0	0	1	0	1	1	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0	0	0
1	0	0	1	1	1	0	0	0	0	0
1	0	0	1	1	1	1	0	0	0	0



FIG 20-7-5

FLOW CHART OF THE DMA CONTROLLER STATE MACHINE

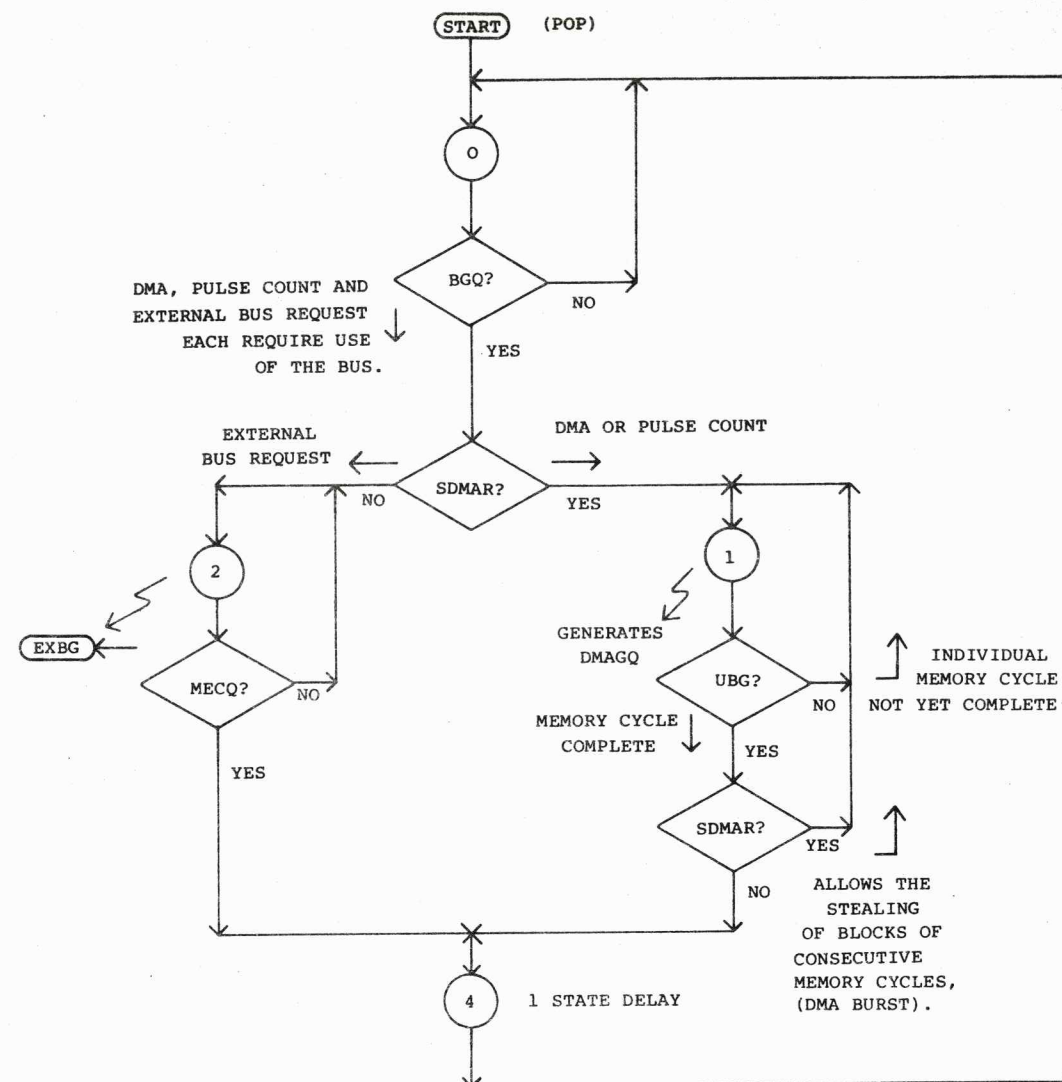


FIG 20-7-6

SECTION 20 (CONTINUED)

Figure 20-7-6 is a practical compilation of the flow chart segments for the individual states. This figure should be considered the actual flow chart of the DMA Controller State-Machine.

Figure 20-7-7 illustrates the relationship between the Bus Controller and the DMA Controller. This drawing will make more sense once the reader has familiarized himself with the DMA and Pulse Count portions of the Bus Controller.

SECTION 21

Figure 21 explains the conventions used in the ASM Charts of the Instruction Controller and of the Bus Controller, and is self-explanatory.

RELATIONSHIP BETWEEN THE BUS CONTROLLER AND THE DMA CONTROLLER

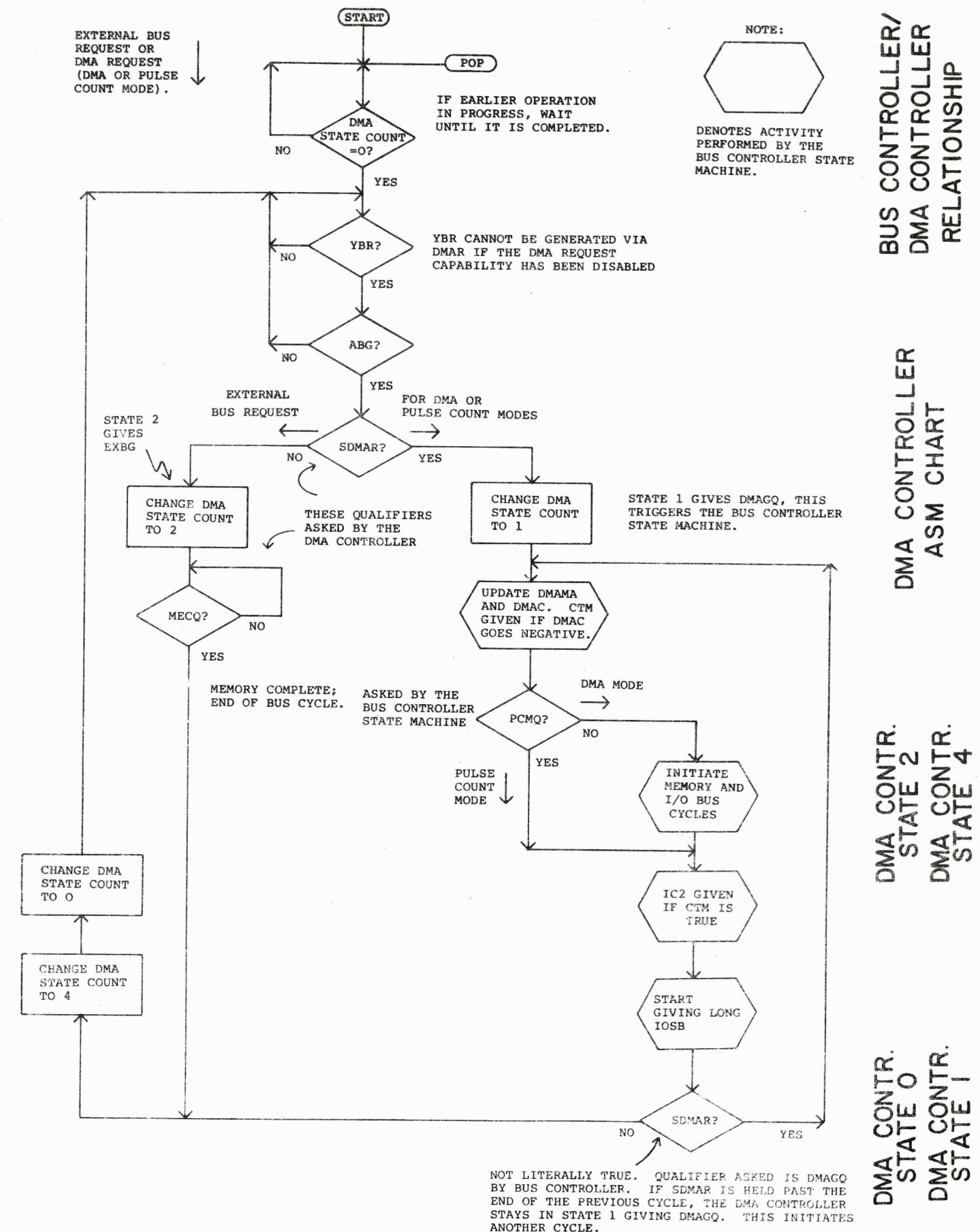
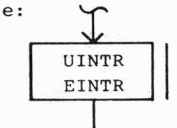


FIG 20-7-7

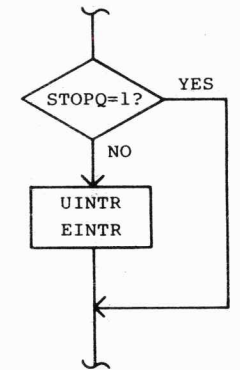
INTERPRETING THE IOC ASM CHART

1. What we usually refer to simply as a state ("state 2 for a PLACE instruction") is generally a coincidence of that particular state-count and some other qualifiers (each state machine has its own collection). The most precise way to refer to a location on an ASM chart is to indicate which state machine, and which instruction or operation. Some states (0) within a particular machine are completely independent of all qualifiers except the state count itself and those particular qualifiers that effect the specific instructions decoded within the state. States are indicated by circles with numbers in them: ①. Instruction group information is prominently displayed next to sections to which it pertains.
2. Each state represents a 02 pre-charge and 01 decode in the ROM. The ASM chart represents what is decoded from the ROM in the various states; it does not necessarily represent end-results that occur simultaneously. If, for instance, two instructions decoded in the same state have different delays coming from the ROM, then they do not result in simultaneous activity, even though they are drawn as being in the same state.
3. Rectangular boxes (DMP C) denote micro-instructions. Diamonds (MECQ=1?) denote qualifiers affecting the decoding of micro-instructions within a state.

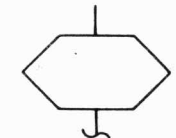
Some micro-instructions are accompanied by a vertical bar to the right of their enclosing rectangle:



This is a short hand notation for denoting that (all) the micro-instructions in that rectangle are conditional upon STOPQ, thus:

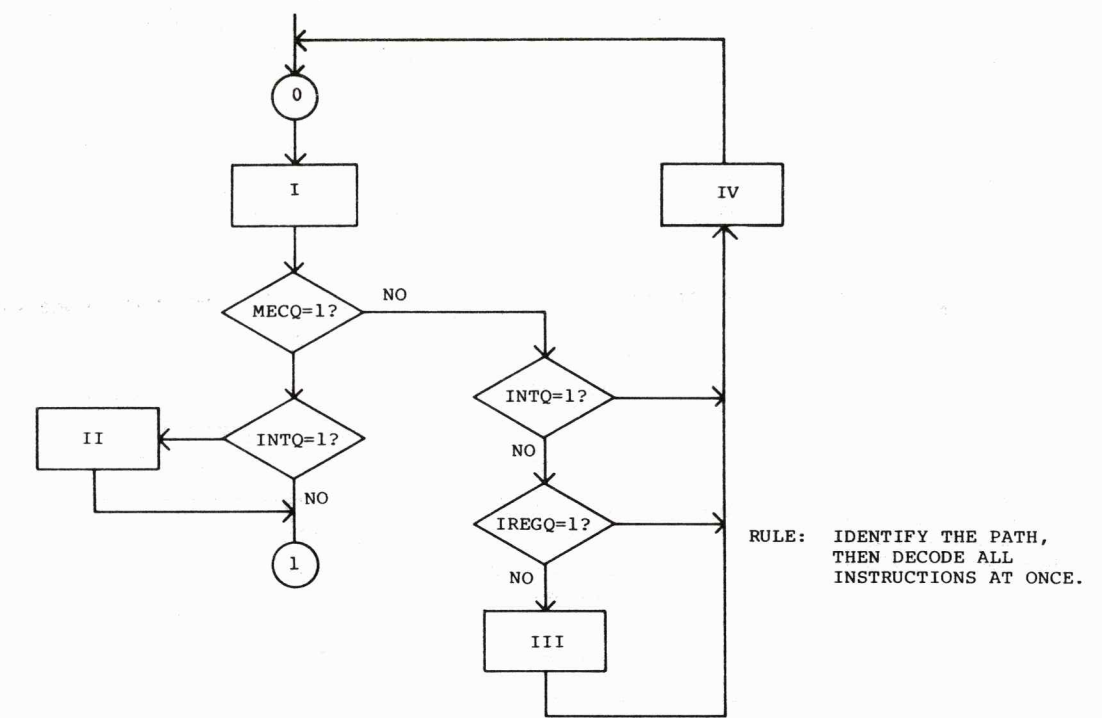


The symbol



is used to denote various different things, depending upon the flow chart. In each instance a note on the flow chart indicates the meaning. The symbol is most widely used to denote activity controlled be a different state-machine, or, as a shorthand notation to denote an often used but awkwardly drawn segment of flow chart.

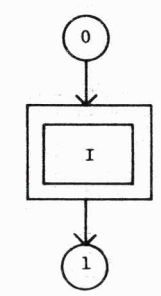
4. All activity within a state is decoded and initiated (its delay is begun) at the same time. The fact that a state is shown as a sequential arrangement of boxes and diamonds does not imply sequential activity; the entire state is decoded simultaneously. For example, state 0 of the Instruction Controller is represented below:



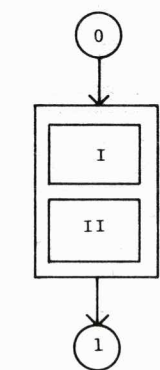
RULE: IDENTIFY THE PATH, THEN DECODE ALL INSTRUCTIONS AT ONCE.

Another way to represent the same activity is illustrated below. We don't draw the ASM chart that way because of the increased size and because of problems in achieving connectedness. Also, overall algorithmic process would be hard to see; the more compact notation results in a more effective visual outline. Within a state however, the expanded notation is often less confusing as it more closely represents the actual way things are done.

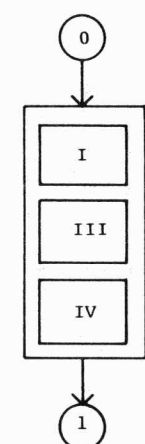
If MECQ=1 and INTQ=0, then:



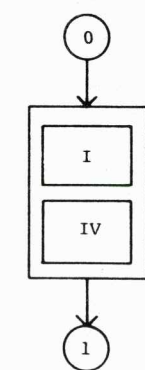
If MECQ=1 and INTQ=1, then,



If MECQ=0, INTQ=0 and IREGQ=0, then:



If MECQ=0, and either of INTQ or IREGQ=1, then:



5. Within a state, related instructions are grouped together in the same box solely for the sake of algorithmic clarity.
6. Because of limitations on transistor device size, and the large capacitances of the IDA lines, two consecutive SET IDA'S are required to ensure that IDA lines assume their proper final values.

FIG 21

INSTRUCTION CONTROLLER ASM CHART OVERVIEW



125

SECTION 22 (CONTINUED)

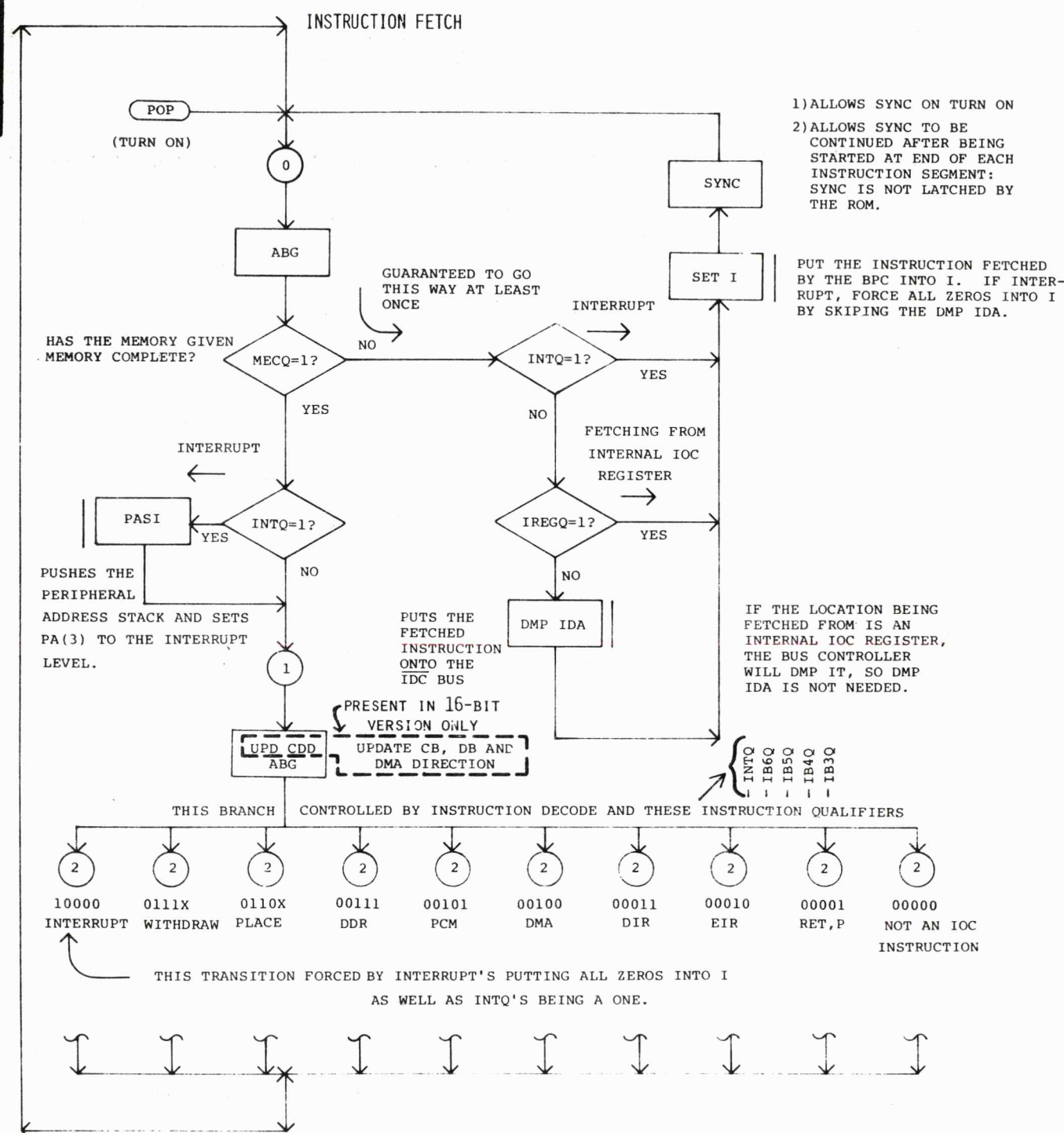


FIG 22-2

Figure 22-2 illustrates the Instruction Fetch and Fan-out portion of the Instruction Controller's ASM chart. Observe that this also constitutes the turn-on sequence in the Instruction Controller.

Recall that it is the BPC that actually initiates and performs the instruction fetch. The Instruction Controller listens to the fetched instruction and copies it into the I register. A fan-out then occurs according to the nature of the fetched instruction.

An interrupt request which has been granted causes some special activity. In the first place, the fetched instruction bit pattern is ignored, and all zeros are forced into its place. This is part of the mechanism that selects the Interrupt segment of the Instruction Controller's ASM chart. The second activity is that at the conclusion of the instruction fetch the micro-instruction PASI is issued. This pushes the interrupting select code onto the Peripheral Address Stack and affects the Interrupt Controller in the previously described manner.

The difference between the 15 and 16-bit versions concerns the CB, DB and DMAD registers. In this regard, the 16-bit version issues UPD CDD in state one. UPD CDD endures the usual one-state delay. It is used by Instruction Decode, and by the time the delay is over the I register has been set-up. UPD CDD and the bit pattern in the I register are gated together to determine which, if any, of UPD DMAD, UPD CD and UPD DB should be issued. The effect of all this is to assist the machine-instructions that control the CB, DB and DMAD registers.

There is a bug concerning instruction fetches of IOC machine-instructions from registers within the IOC. (Also see Figure 23-4 in this connection.) The bug is that the IOC will not recognize the fetched instruction, and will treat it as if it were a non-IOC instruction.

The reason for this is that state one of "Read" segment of Figure 23-4 does not contain a "DMP

INTERNAL REGISTER" micro-instruction. What happens is this. During the instruction fetch segment of Figure 22-2 the Bus Controller (see Figure 23-4) dumps the register being read into 0, and sends it out with two consecutive SET IDA's. Only one DMP is done, however. Now, the sequence in Figure 22-2 does two SET I's; one for each SET IDA being done by whoever is providing the instruction bit pattern to the IDA Bus. The intent was that since the IOC is doing all the talking, anyhow, to let the "DMP INTERNAL REGISTER" match up with the SET I, and "pass the instruction under the table," as it were. The data does go into the 0 register, and the two SET IDA's properly drive the IDA Bus. Unfortunately, however, the Instruction Controller is not listening to the external IDA Bus, it is doing a SET I with no SET IDA and no second DMP of the addressed register. The result is to set the I register to all zeros. (It appears the engineer was attempting to avoid a SET IDA/DMP IDA situation in conjunction with the original DMP of the internal register. That would confuse the internal IDA Bus, unless the 0 register were set up in advance. He simply overlooked the fact that the second SET I is necessary. In the corresponding situation in the BPC there is only one state machine, and it is all done in one state. In the IOC there are two state machines involved, and several states, so this was an easy oversight to make.)

"NOT AN IOC INSTRUCTION"
SEGMENT OF THE INST. CONTROLLER ASM CHART
NOT AN IOC INSTRUCTION

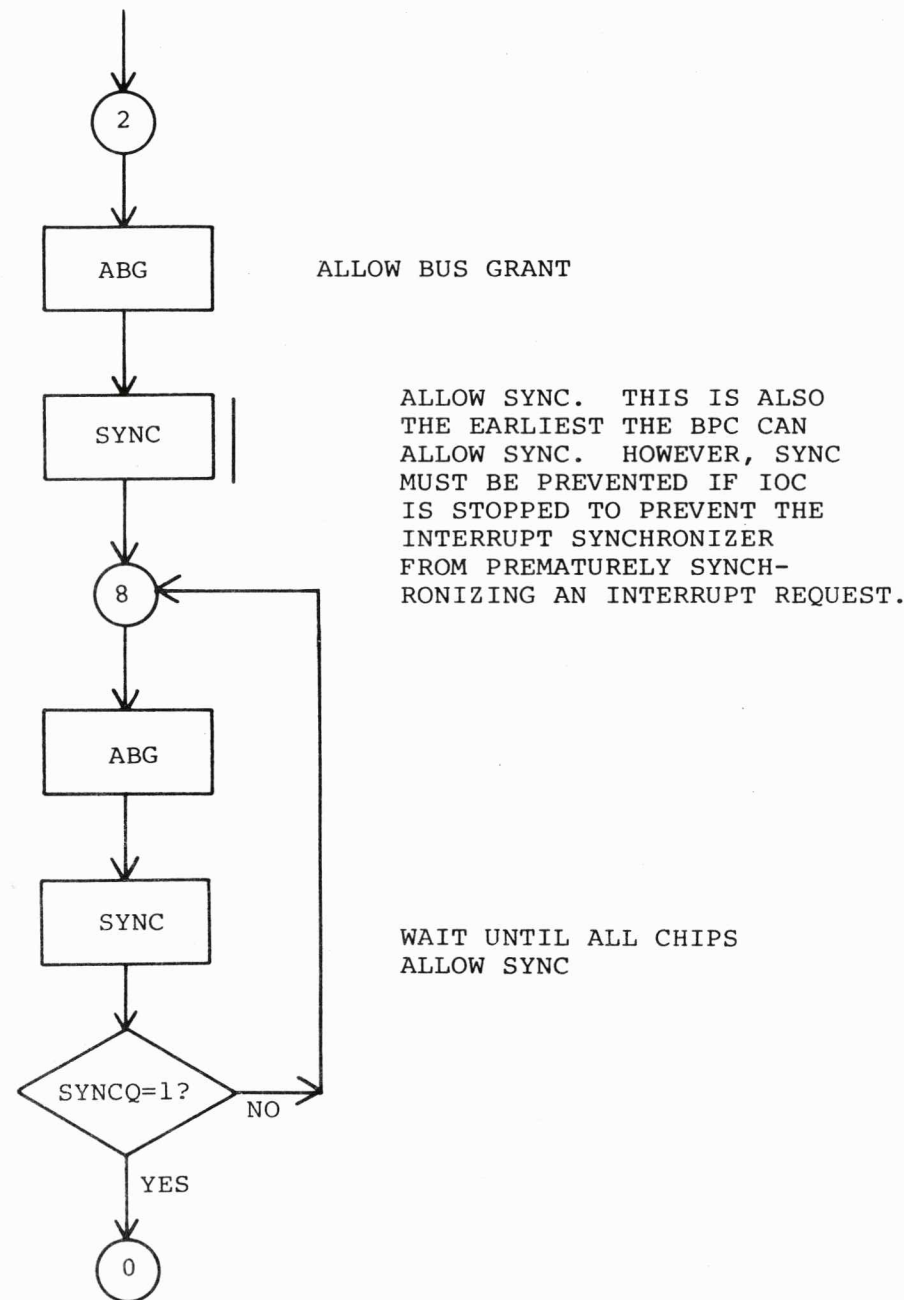


FIG 22-3

SECTION 22 (CONTINUED)

Figure 22-3 illustrates the segment of the Instruction Controller's ASM chart that is reached when the fetched instruction does not pertain to the IOC. Basically, it is simply an idling loop in which the Instruction Controller waits until all the chips in the system have agreed to allow SYNC to go true.

"RET,P" SEGMENT OF THE INST. CONTROLLER ASM CHART
RET,P

(RETURN FROM AN INTERRUPT SERVICE ROUTINE)

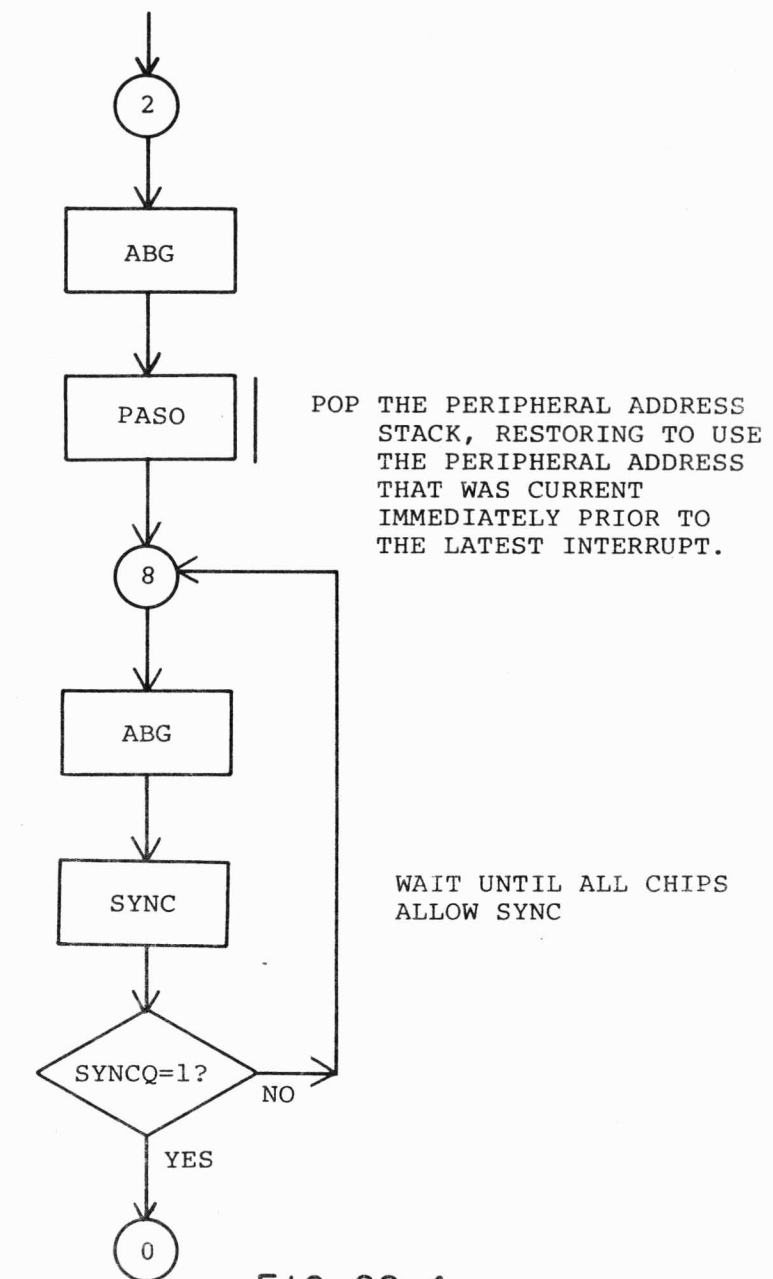


FIG 22-4

Figure 22-4 shows the RET,P segment of the Instruction Controller's ASM chart. There are two things that happen in this segment. First, the micro-instruction PASO is issued to undo the stacking effects caused by the previous interrupt. That is, it pops the Peripheral Address Stack and restores the previous select code-in-use. Next, the remainder of the segment serves as an idle loop in

which to wait until all chips in the system have agreed to allow SYNC to go true.

100

"EIR" SEGMENT OF THE INST. CONTROLLER ASM CHART

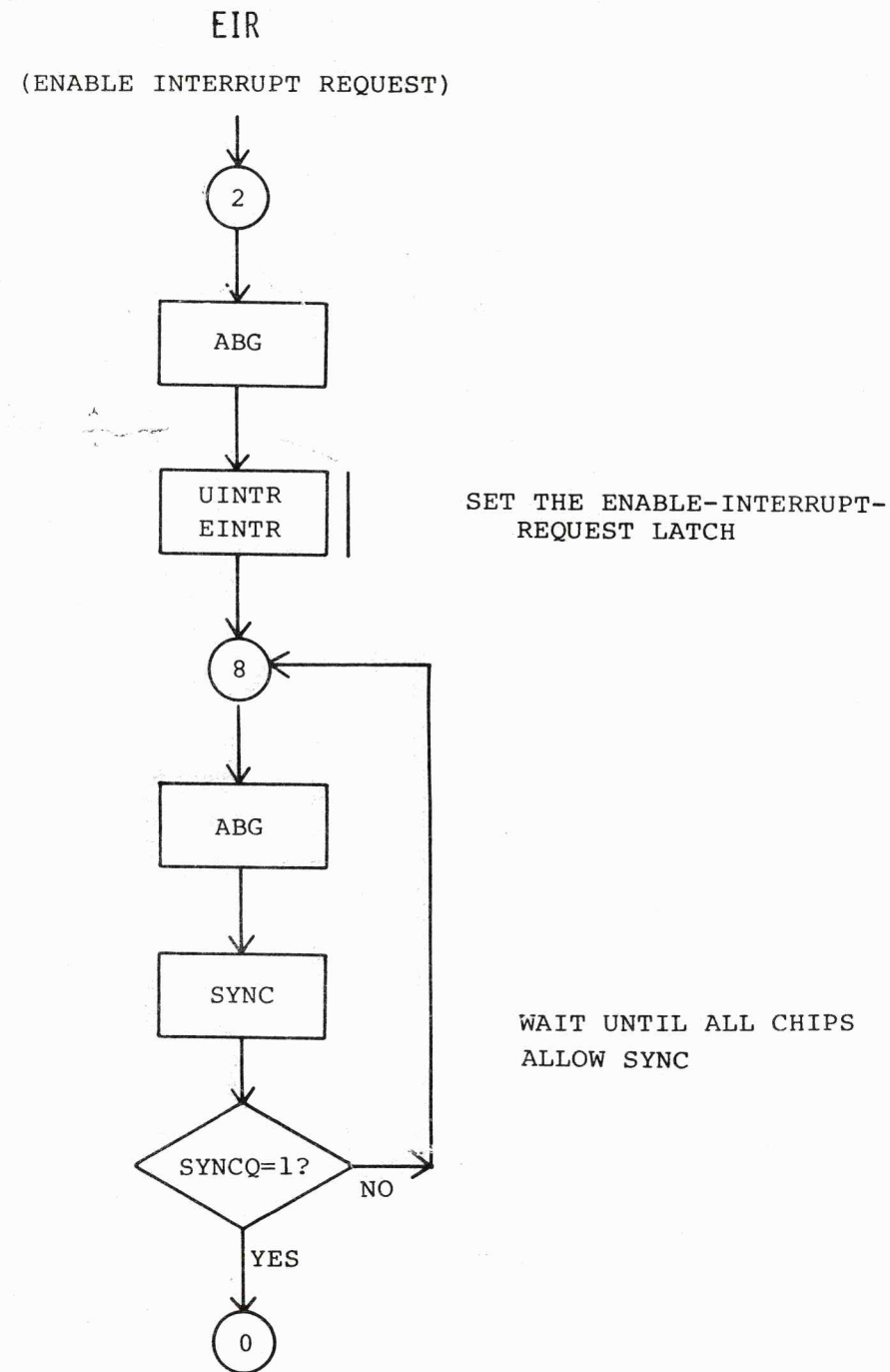


FIG 22-5

SECTION 22 (CONTINUED)

Figure 22-5 shows the segment of the Instruction Controller's ASM chart that corresponds to the Enable Interrupt Request (EIR) machine-instruction. The central activity of this segment is to issue the micro-instructions UINTR and EINTR, which result in setting the Enable Interrupt Request Latch. The remainder of this segment is the usual

idle loop in which to wait until SYNC goes true.

"DIR" SEGMENT OF THE INST. CONTROLLER ASM CHART

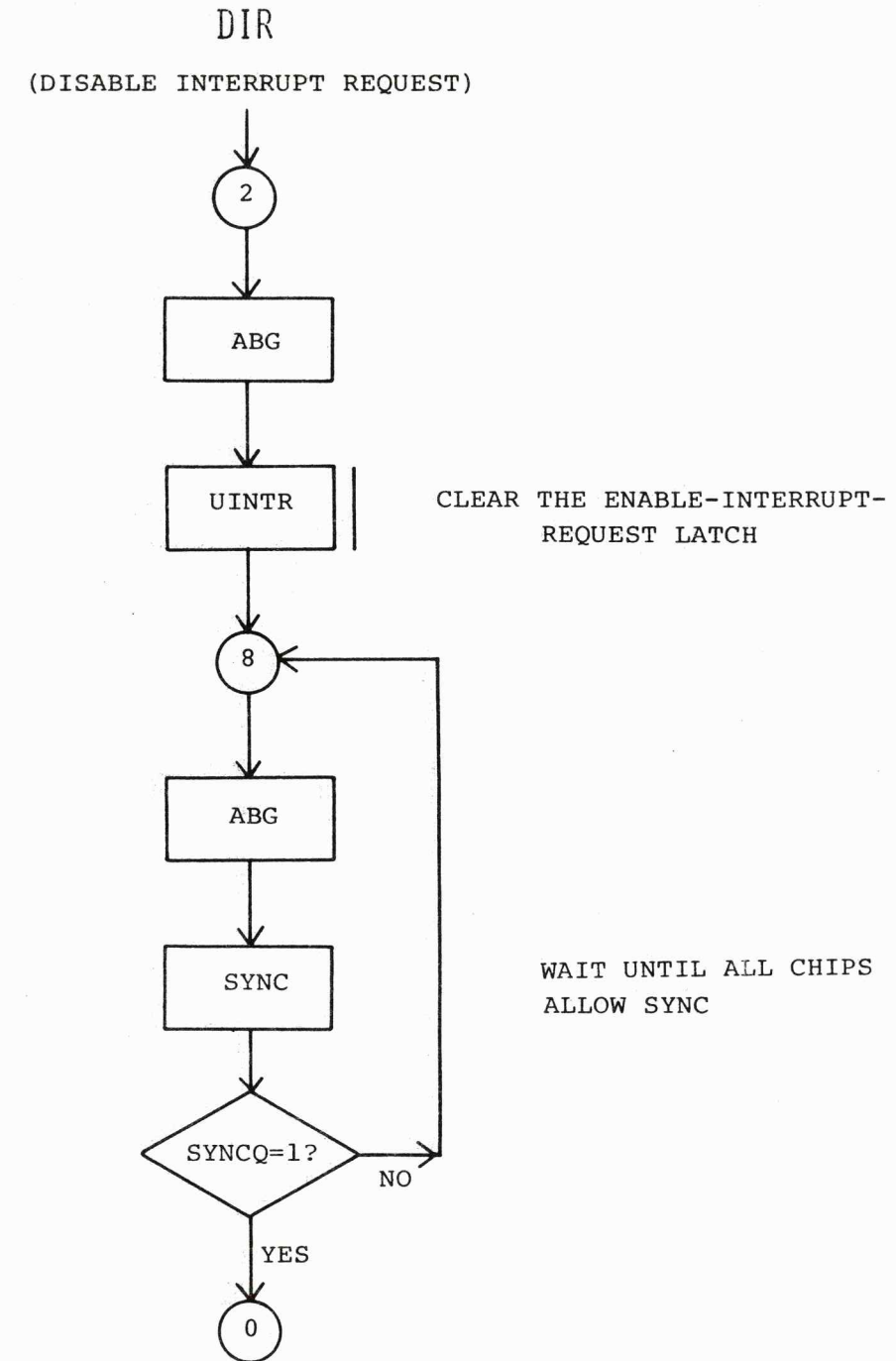
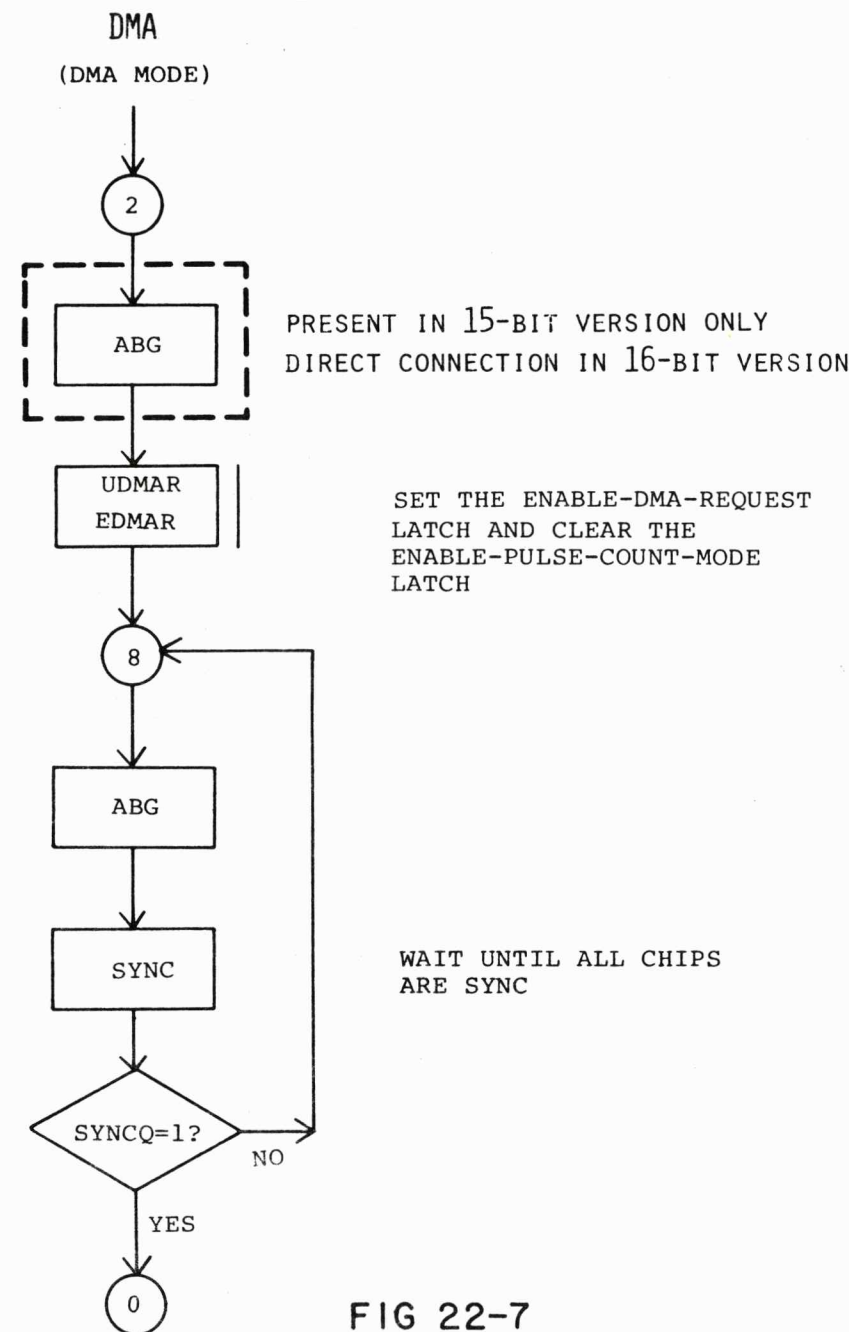


FIG 22-6

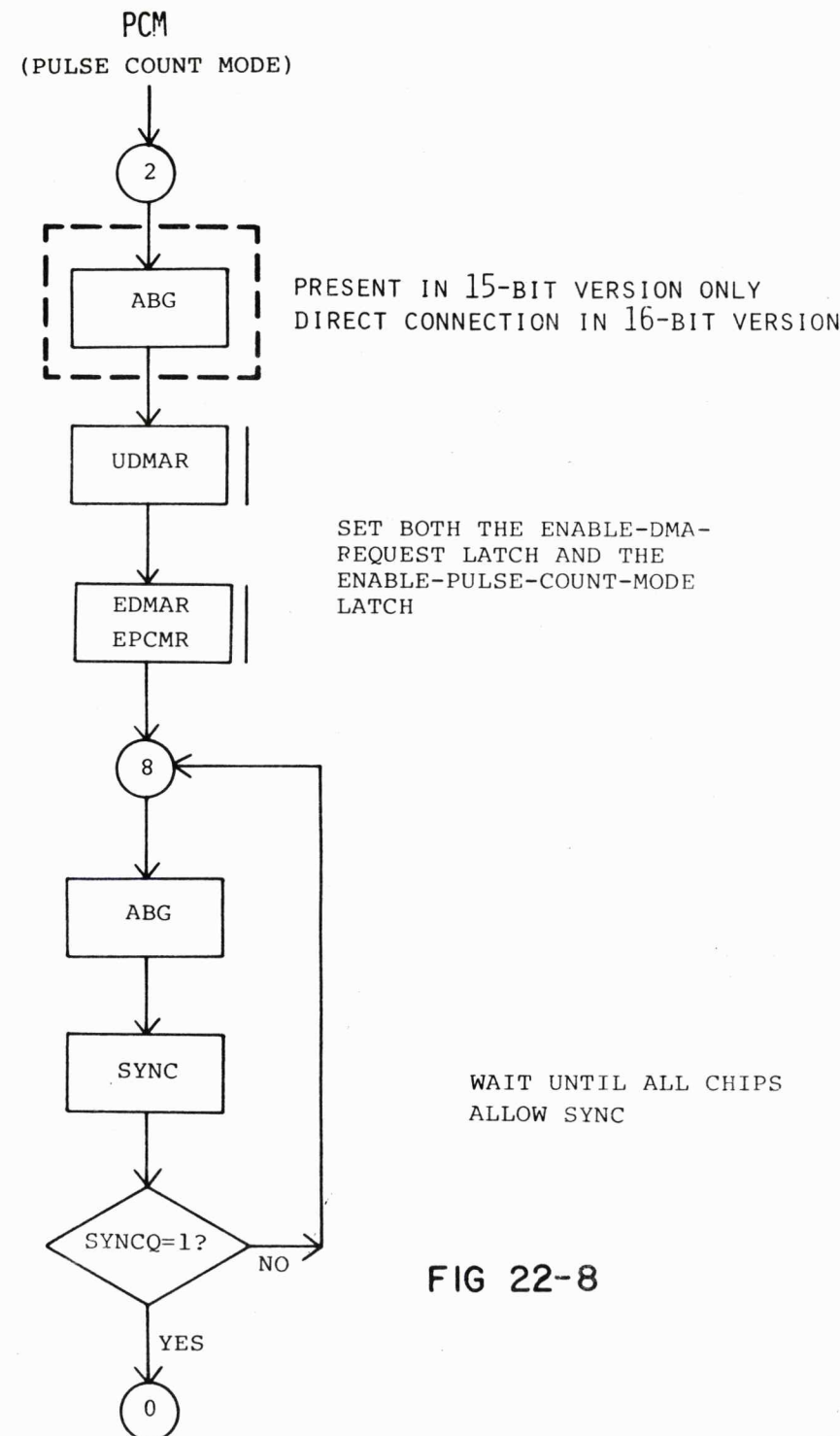
Figure 22-6 shows the segment of the Instruction Controller's ASM chart that corresponds to the Disable Interrupt Request (DIR) machine-instruction. This segment differs from the EIR segment only in that state two issues UINTR without a corresponding EINTR. An inspection of the Enable Interrupt Request Latch circuitry will reveal

that this combination of instructions clears the latch. After the latch is cleared there is the customary wait until SYNC is true.

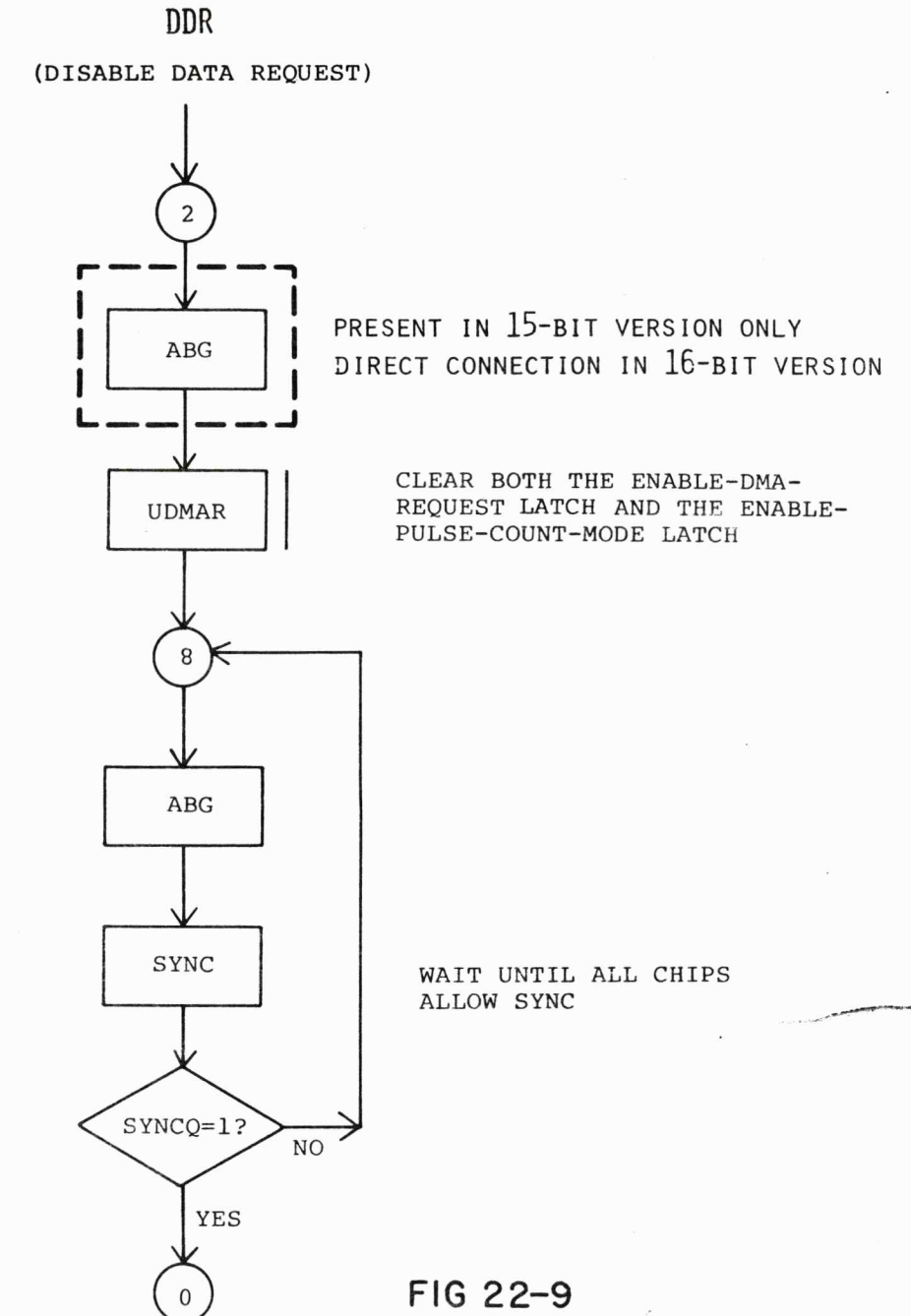
"DMA MODE" OF THE INST. CONTROLLER ASM CHART



"PCM" SEGMENT OF THE INST. CONTROLLER ASM CHART



"DDR" SEGMENT OF THE INST. CONTROLLER ASM CHART



SECTION 22 (CONTINUED)

Figure 22-7 shows the segment of the Instruction Controller's ASM chart that corresponds to the DMA Mode (DMA) machine-instruction. This segment first sets the Enable DMA Request Latch by issuing the micro-instructions UDMAR and EDMAR. Next, the segment waits until SYNC goes true.

The difference between the two versions concerns whether or not a Bus Grant is allowed during the time the latch is actually changing.

To allow a Bus Grant during that time causes a bug. This bug is described in the N-MOS II Processor Manual.

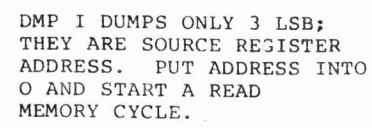
Figure 22-8 shows the segment of the Instruction Controller's ASM chart that corresponds to the Pulse Count Mode (PCM) machine-instruction. This segment establishes the Pulse Count Mode by setting both the Enable DMA Request Latch and the Enable Pulse Count Mode Latch. This is done by

issuing a micro-instruction UDMAR in conjunction with EDMAR and EPCMR. Next, there is the usual wait until SYNC is true.

The difference between the two versions concerns the same bug relating to DMA requests, as described in the previous section.

Figure 22-9 shows the segment of the Instruction Controller's ASM chart that corresponds to the Disable Data Request (DDR) machine-instruction.

The main activity of this segment is to clear both the Enable DMA Request Latch and the Enable Pulse Count Mode Latch. This is done by issuing the micro-instruction UDMAR without any corresponding Enable micro-instruction. The effect of this is to disable the DMA Request Line and cancel the Pulse Count Mode. Next, there is the usual wait until SYNC is true.

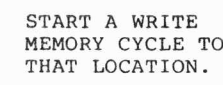


UPDATE THE
POINTER REGISTER

PUT CONTENTS OF
SOURCE REGISTER
INTO W.

DUMP THE ADDRESS
CONTAINED IN THE
POINTER REGISTER.
THE DUMPED BIT
PATTERN IS THE
ADDRESS OF THE
DESTINATION WORD.

FIG 22-10-S



```

EBYT RESULTS IN
BYTE UNLESS STP
IS TRUE.

```

OUTPUT THE DATA TO
BE WRITTEN. DMP W
TAKES WORD/BYTE
OPERATION INTO
ACCOUNT.

WAIT UNTIL
WRITE CYCLE
IS COMPLETE

ALLOW SYNC,
BUS GRANT

Figures 22-10-S and -F are the Instruction Controller ASM chart segments corresponding to "Place" machine-instructions. The function of a "Place" machine-instruction is to extract the contents of some source register and place them into a location in memory specified by a pointer register. The source register can be any of the first eight address-
es, and the pointer can be either the C or D register.

The first thing that is done is to obtain the address of the source register. It is encoded in the bottom three bits of the machine-instruction in the I register. The address is obtained with the micro-instruction DMP I; DMP I operates on only the three least significant bits of the I register. The address is put into the 0 register and a read memory cycle is initiated. The contents of the source register will be put into W. Meanwhile, the value of the associated pointer register is updated. This is done with a UPD C or UPD D, as is appropriate. The choice is based upon the C/D bit of the machine-instruction. Whether the update instruction causes an increment or decrement is controlled by the effect of signals from Instruction Decode upon C/D Register Control.

Once the data to be transferred is in W, DMP C or DMP D, or one of their variants, is used to generate the proper word or byte address to which the data will eventually be written. The result of the DMP of the pointer register is set into the 0 register and a write memory cycle initiated.

The remainder of the segment is the transmission of the data (which is in W). W is set into the 0 register and repeatedly sent via SET IDA until a Memory Complete is received.

During the write memory cycle just described the BYTE line is controlled to reflect whether the transmitted item is an entire word or just a single byte.

There are two areas of difference between the 15 and 16-bit versions. The first of these concerns the DMP C and DMP D micro-instructions.

"PLACE" SEGMENT OF THE INST. CONTROLLER ASM CHART

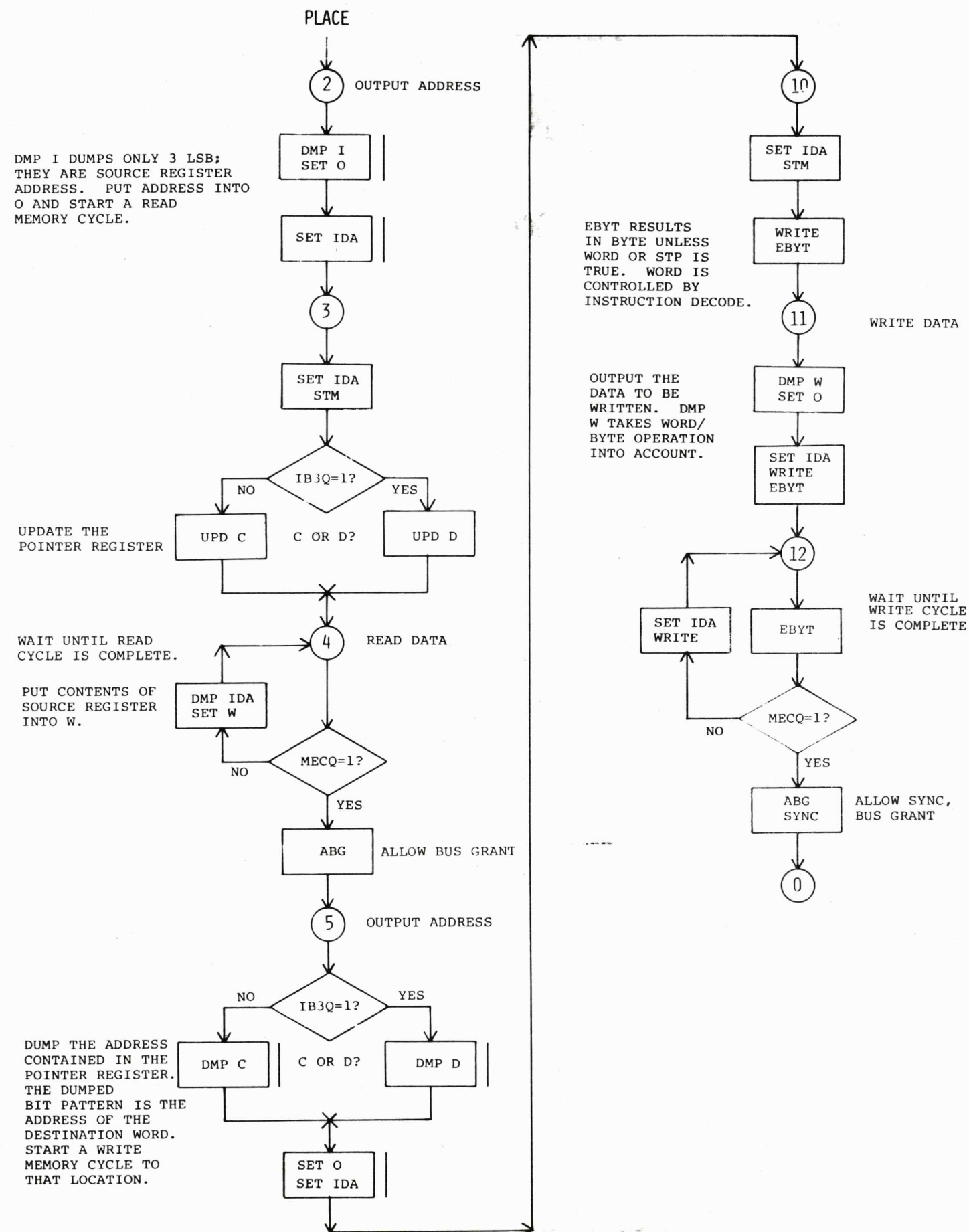


FIG 22-10-F

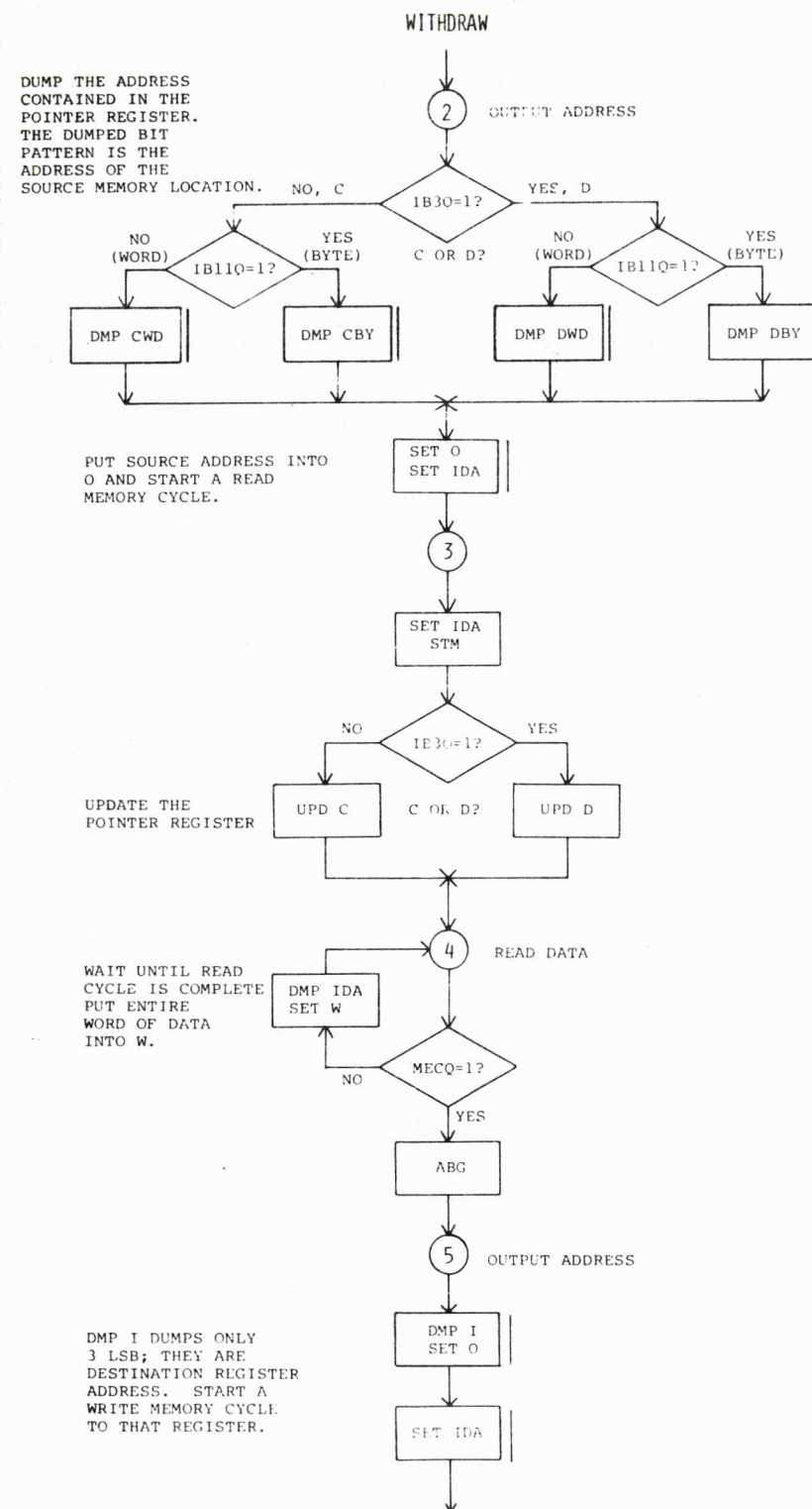
SECTION 22 (CONTINUED)

In the 16-bit version these micro-instructions are implemented as four separate micro-instructions. There is a DMP Word or DMP Byte micro-instruction corresponding to each of C and D, and the flow charting explicitly issues the correct micro-instruction for the situation at hand, based on qualifier information. The difference between the two types of instructions is this. The word oriented instructions simply dump the C or D register for what it is, as a 16-bit quantity. The byte oriented instructions output a 17-bit address with either CB or DB as the most significant bit, while the least significant bit of C or D is used to control the BL line.

The second difference between the two versions is in the way BYTE is handled. In the 16-bit version BYTE is explicitly controlled by the flow chart. In the 15-bit version the micro-instruction EBYT is issued by the ROM without regard for whether BYTE should be issued or not. That decision is made at the location of the driver for BYTE, according to the signal WORD from Instruction Decode.

PLACE

"WITHDRAW" SEGMENT OF THE INST. CONTROLLER ASM CHART



SECTION 22 (CONTINUED) FIG 22-II-S

Figures 22-II-S and -F are the Instruction Controller ASM chart segments for the various "Withdraw" machine-instructions. Except for interchanging the order of the major operations, these machine-instructions are quite similar to the "Place" machine-instructions just described, including the details of the difference

between the 15-bit and 16-bit versions. The purpose of the various "Withdraw" machine-instructions are to extract a word from memory, according to the value of a pointer register, and then install the entire word, or a byte of that word, into a destination word, which as before, is encoded in the bottom three bits of the machine-

"WITHDRAW" SEGMENT OF THE INST. CONTROLLER ASM CHART

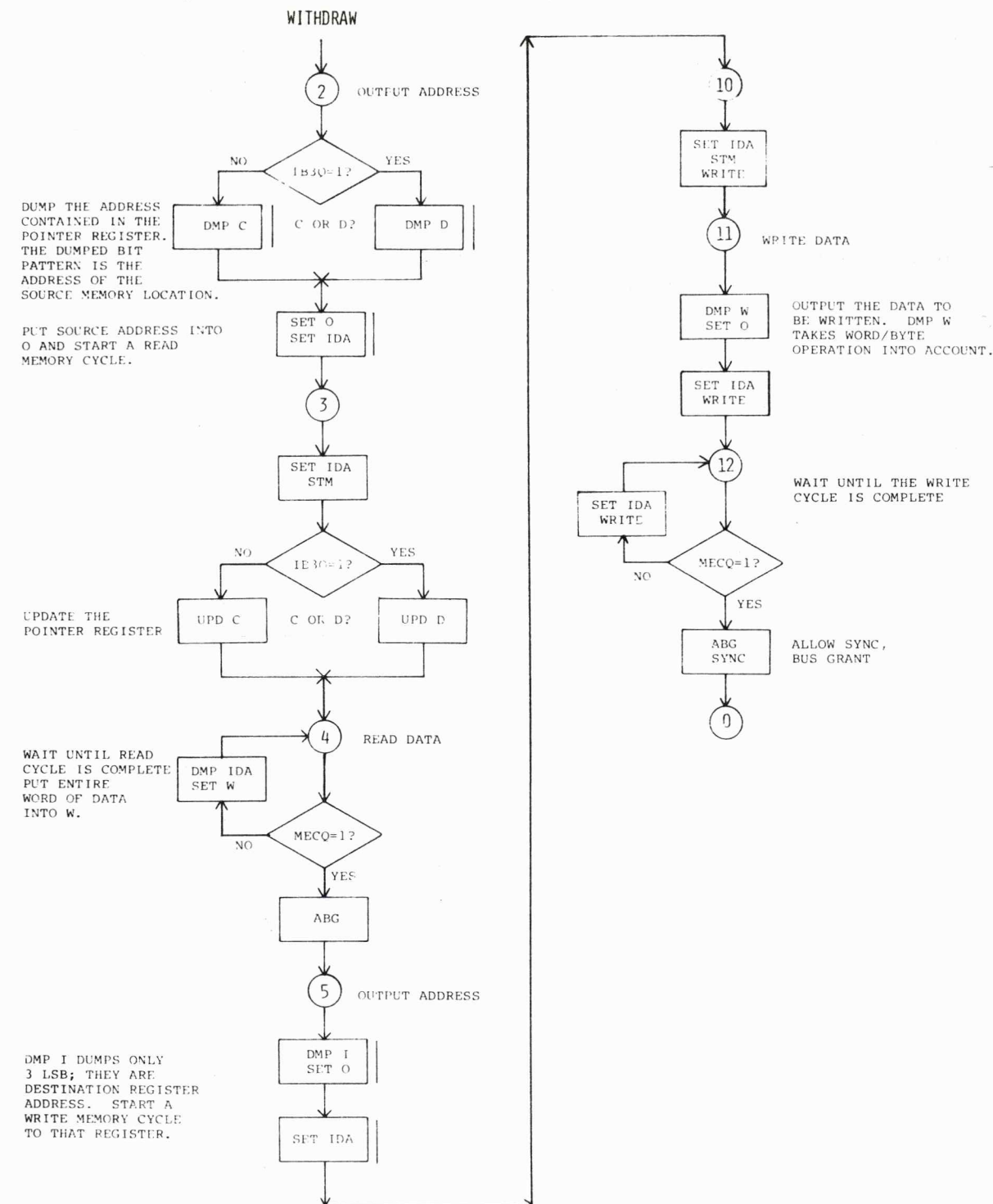


FIG 22-II-F

instruction itself. The first thing that is done is to dump the pointer address. As in the case of a "Place" machine-instruction this results in a word-oriented or byte-oriented address. A read memory cycle is initiated and the data is temporarily stored into the W register. After the memory cycle has been started the

value of the pointer register is updated. Once the memory cycle is completed a write memory cycle to the destination address is initiated. This is done by using the lower three bits of the I register as the address, and the contents of W as the data. The BYTE line is not involved in "Withdraw" machine-instructions.

"INTERRUPT" SEGMENT OF THE INST. CONTROLLER ASM CHART

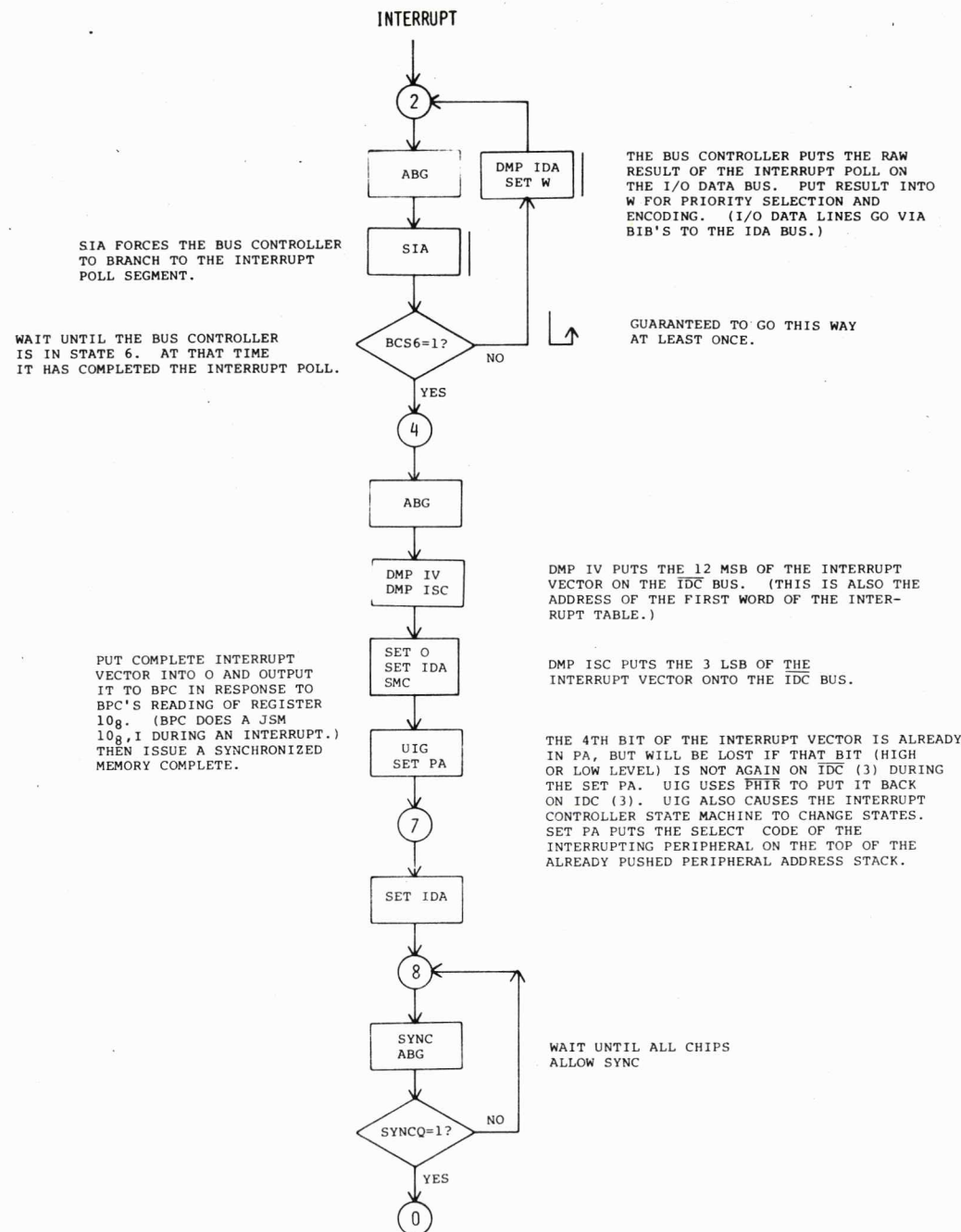


FIG 22-12

SECTION 22 (CONTINUED)

Figure 22-12 shows the segment of the Instruction Controller's ASM chart that is invoked when an interrupt occurs. The purpose of this segment of flow charting is to create the interrupt vector transmitted in response to the BPC's interrogation of the IV register (during the BPC's response to an interrupt).

The first thing that is done is to force the Bus Controller to do an interrupt poll. The performance of the interrupt poll by the Bus

Controller is described elsewhere. However, the results of that process are as follows. The response of the peripheral to the interrupt poll is felt on that portion of the IDA Bus that is internal to the hybrid. While waiting for the completion of the interrupt poll the Instruction Controller does a repeated DMP IDA/SET W to put the response of the interrupt poll into W. The micro-instruction SIA is used to force the Bus Controller to do the interrupt poll. Com-

OVERVIEW OF THE BUS CONTROLLER ASM CHART

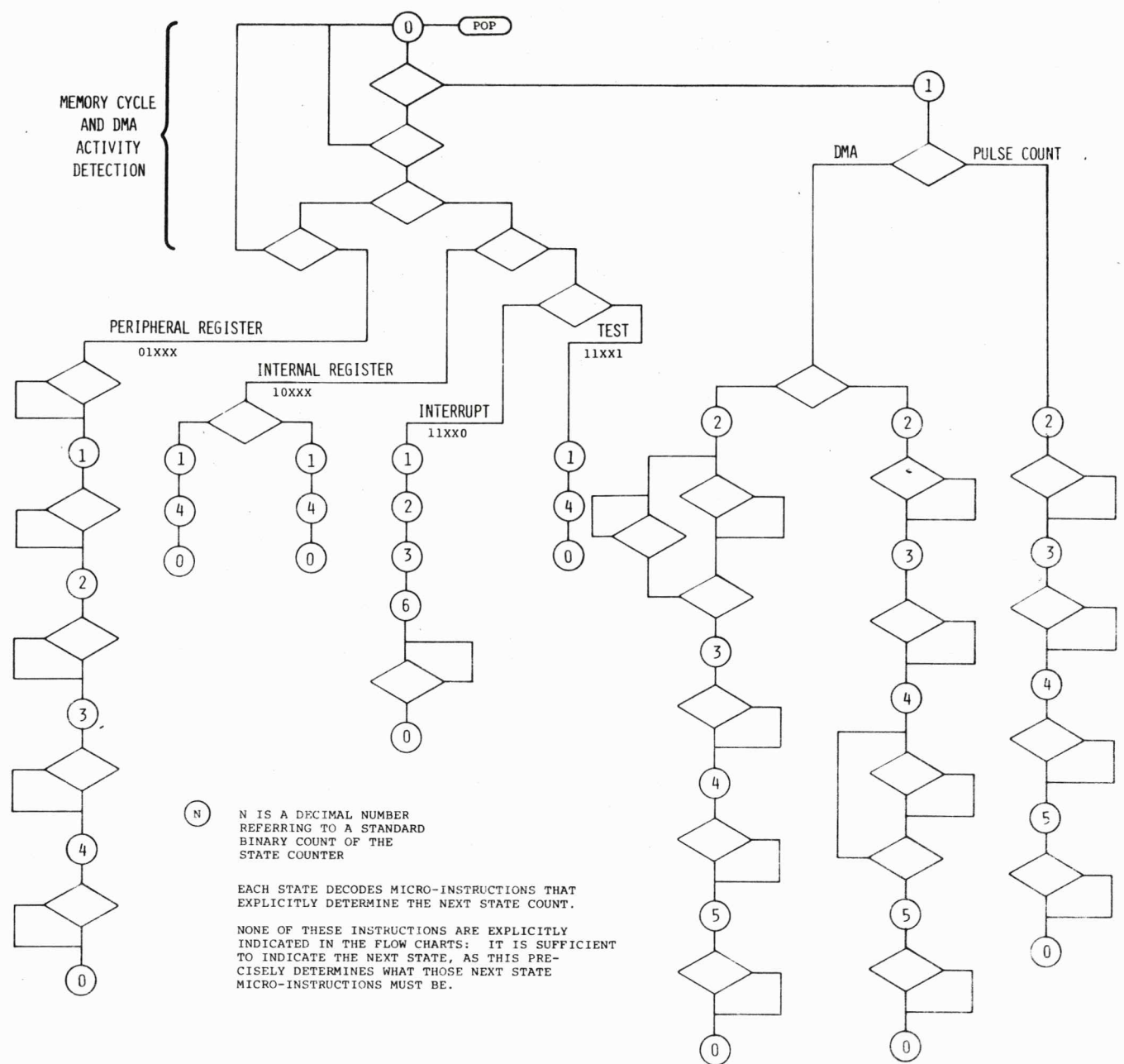


FIG 23-1

pletion of the interrupt poll is signaled to the Instruction Controller by a qualifier representing the state-count of the Bus Controller.

The next thing that is done is to form the actual interrupt vector into the 0 register. This is done by a series of simultaneous dumps of the various portions of the interrupt vector. The top twelve bits come from the IV register, and are obtained by a DMP IV. The fourth bit comes from the Interrupt Control-

ler, and is obtained by the issuance of the micro-instruction UIG. UIG not only updates the state of the Interrupt Controller, but applies the interrupt level to bit 3 of the IDA Bus. The bottom three bits of the interrupt vector are obtained by the micro-instruction DMP ISC. The DMP ISC extracts the encoded result obtained from putting the interrupt poll information into W.

At this time the bottom four bits of the interrupt vector are also

BUS CONTROLLER ASM CHART OVERVIEW

INTERRUPT

WITHDRAW

WITHDRAW



Next, the completed interrupt vector, which is now in the 0 register, is sent to the BPC in response to the read memory cycle that was initiated for register 10₈. After that, there is the customary wait until SYNC is true.

Figure 23-1 illustrates the extent of the ASM chart for the Bus Controller.



SECTION 23 (CONTINUED)

Figure 23-2 shows the Memory Cycle and DMA Request Detection segment of the Bus Controller's ASM chart. The purpose of this segment is to set the Bus Controller in motion whenever one of two things occurs. These two things are either a memory cycle or a DMA request (of which the latter could be either for genuine DMA operation or for Pulse Count Mode operation). The nature of the detected operation is determined, and a branch is executed to the appropriate segment of Bus Controller activity.

Consider the case of a memory cycle. Memory cycles come in different flavors. Part of the purpose of Address Decode is to generate qualifiers that correspond not only to the address, but to the *type* of memory cycle the address identifies. The Bus Controller uses these qualifiers, shown on the drawing, to branch to the various separate segments corresponding to the different types of memory cycles. These types are:

Peripheral Registers (The original reference was to one of registers four through seven.)

Internal Registers (The reference is to an actual register within the IOC.)

Interrupt (Do an interrupt poll in response a reference to register 10₈ while INT is low.)

Test (Used only in testing the IOC, and is an ERA Mode operation allowing a SET I.)

In the event that a DMA request is received the address in DMAMA is sent out and the counter registers associated with DMA and the Pulse Count Mode are updated. Start Memory is not yet issued, however; a Pulse Count Mode operation does not involve an actual memory cycle. A branch is then made to the appropriate segment: DMA or Pulse Count Mode.

Figure 23-3 illustrates the peripheral register segment of the Bus Controller's ASM chart. The purpose of this segment is to generate a standard I/O Bus Cycle. This does not involve handling the data as much as it does managing the control signals.

Activities that occur throughout most of this segment are the issuance of Buffer Enable (BE) and the controlling of \overline{DOUT} according to whether this is to be a read or write I/O Bus Cycle. $\overline{IC1}$ and $\overline{IC2}$ are also issued according to the register referenced in the original memory cycle.

In the case of a read operation the peripheral supplies the data to the IOD Bus. The IOD Bus is connected to the IDA Bus because Buffer Enable has been issued. The buffers will be enabled in the proper direction since the micro-instruction DIN is issued during a read.

In the case of a write operation the originator of the memory cycle supplies the data. As before, the IDA Bus and the IOD Bus are connected. But in this case the buffers are enabled in the other direction.

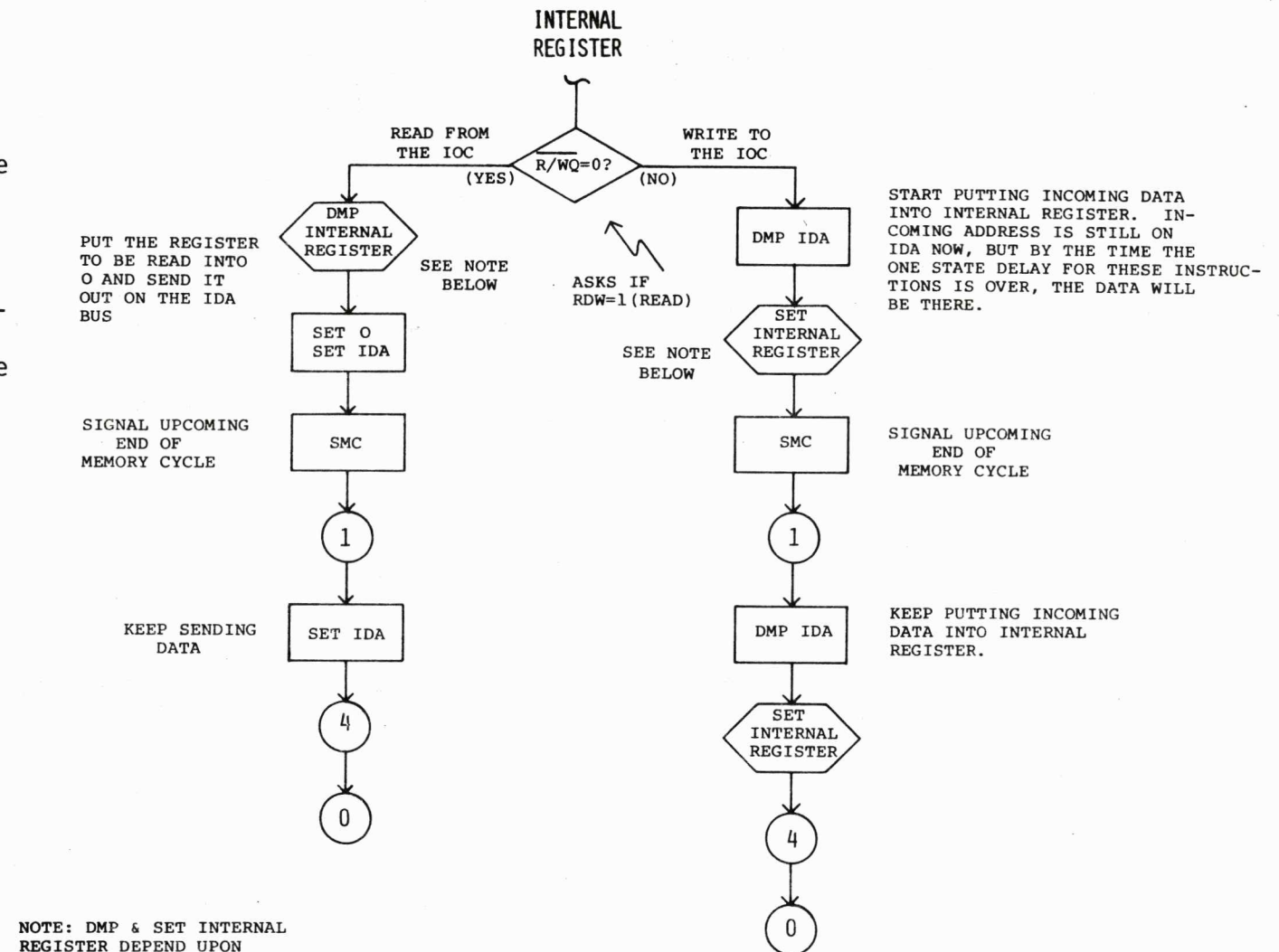
Note that in both these cases the timing is fixed. No qualifiers are employed to allow a variable time-length handshake mode of operation.

Figure 23-4 shows the segment of the Bus Controller's ASM chart that corresponds to a memory cycle referencing an internal IOC register. The purpose of this segment is to respond to such memory cycles.

In the event of a read memory cycle the appropriate internal register is dumped and set into 0. Two consecutive SET IDA's are used to send it onto the IDA Bus.

In the event of a write memory cycle two consecutive DMP IDA's, each associated with a set of the appropriate internal register, are used to capture the data.

"INTERNAL REGISTER" SEGMENT OF THE BUS CONTROLLER ASM CHART



ADBQ	ADBQ	ADBQ	DMP INTERNAL REG.	SET INTERNAL REG.
0	0	0	DMP IV	SET IV
0	0	1	DMP PA	SET PA
0	1	0	DMP W	SET W
0	1	1	DMP DMAPA	SET DMAPA
1	0	0	DMP DMAMA	SET DMAMA
1	0	1	DMP DMAC	SET DMAC
1	1	0	DMP C	SET C
1	1	1	DMP D	SET D

FIG 23-4

INTERNAL
REGISTER

PERIPHERAL
REGISTER

STM & DMAR
DETECTION

INTERRUPT POLL SEGMENT OF THE BUS CONTROLLER ASM CHART

INTERRUPT (POLL)

SET THE PERIPHERAL BIB DIRECTION TO BE FROM THE PERIPHERAL INTO THE IOC, AND ENABLE THE PERIPHERAL BIB'S.

DITTO

DITTO

LIKEWISE DITTO

"ENABLE PRIORITY RESOLVER" TURNS ON THE SELECT CODE PRIORITY RESOLVER CIRCUITRY ATTACHED TO THE W REGISTER. IT IS THIS THAT FORMS THE LEAST THREE SIGNIFICANT BITS OF THE INTERRUPT VECTOR.

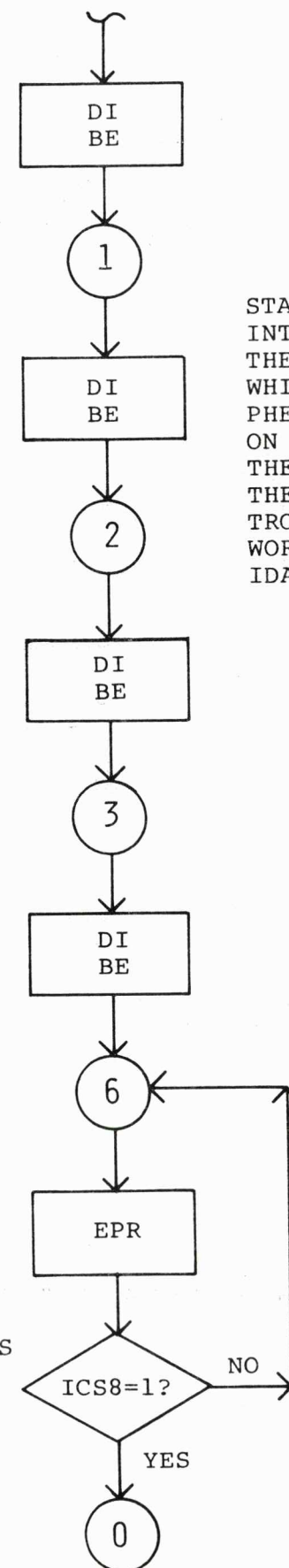


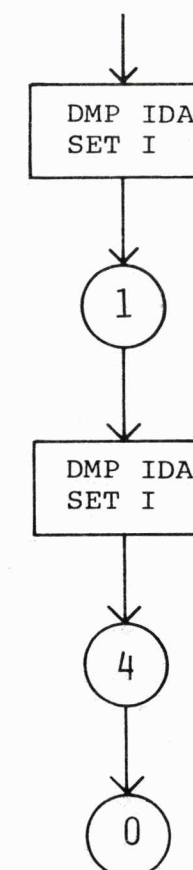
FIG 23-5

STATES 1-3 PUT THE RAW INTERRUPT POLL WORD ONTO THE MOS-LEVEL IDA BUS WHICH IS BETWEEN THE PERIPHERAL AND MEMORY BIB'S ON ONE END, AND THE CHIPS THEMSELVES ON THE OTHER END. THEN THE INSTRUCTION CONTROLLER BRINGS THE POLL WORD INTO THE IOC VIA A DMP IDA-SET W.

WAIT UNTIL THE INSTRUCTION CONTROLLER STATE MACHINE IS IN STATE 8. BY THAT TIME ITS ACTIVITY IS FINISHED AND IT IS WAITING FOR SYNC.

WRITE TO I VIA ERA SEGMENT OF THE BUS CONTROLLER ASM CHART

TEST



THE TEST FUNCTION EXISTS TO ALLOW THE I REGISTER TO BE THE DESTINATION OF A WRITE MEMORY CYCLE. THIS IS ALLOWED ONLY UNDER THE ERA MODE OF ADDRESSING.

FIG 23-6

SECTION 23 (CONTINUED)

Figure 23-5 shows the segment of the Bus Controller's ASM chart corresponding to the performance of an interrupt poll. An interrupt poll is performed at the request of the Instruction Controller (when it issues the micro-instruction SIA). The segment shown in the drawing actually performs the interrupt poll.

To do that, it sets the data transfer direction to be from the peripheral to the IOC, and turns on the BIB's that connect the IOD Bus to the IDA Bus on the hybrid. Then, with the peripheral responding to the conditions requiring it to participate in the poll (INT low and that peripheral requesting interrupt on the level indicated by PAB3), the Bus Controller gives EPR while in a loop until the Instruction Controller has created the actual interrupt vector. The micro-instruction EPR is what is actually used to enable the Priority Resolver to convert the

raw data of the interrupt poll into a useable select code.

Figure 23-6 illustrates the segment of the Bus Controller's ASM chart that corresponds to an ERA mode reference to address 53₂. The purpose of this segment is to provide the means to get something into the I register without doing an actual instruction fetch. It is used only with a write memory cycle in the ERA mode to register 53₂, and then only for test purposes.

Figure 23-7 illustrates that segment of the Bus Controller's ASM chart that deals with genuine DMA operations. Two separate segments are involved, depending upon whether the DMA operation reads from memory and writes to the peripheral, or reads from the peripheral and writes to memory.

We will first consider oper-

"DMA" SEGMENT OF BUS CONTROLLER ASM CHART

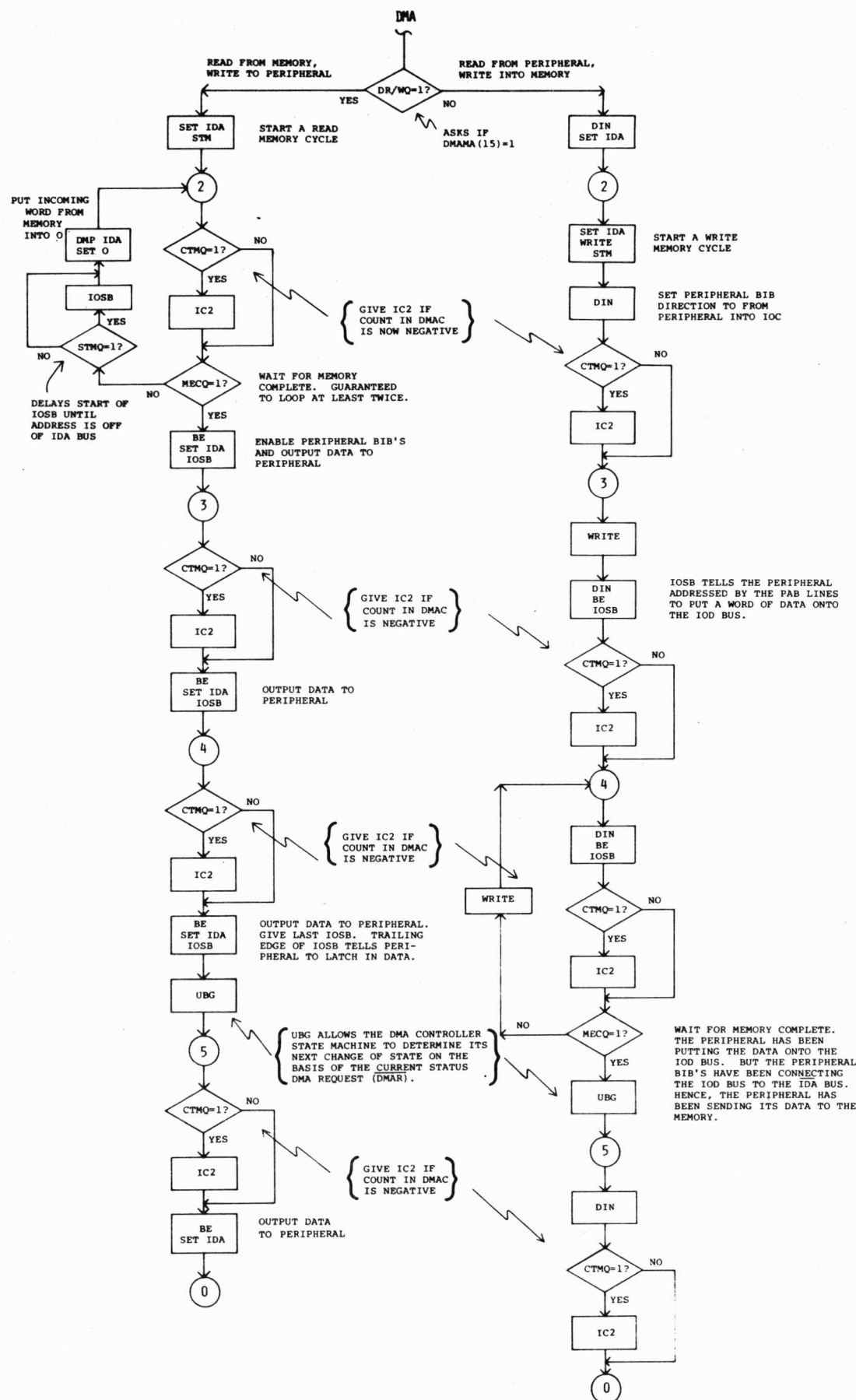


FIG 23-7

"PULSE COUNT" SEGMENT OF THE BUS CONTROLLER ASM CHART

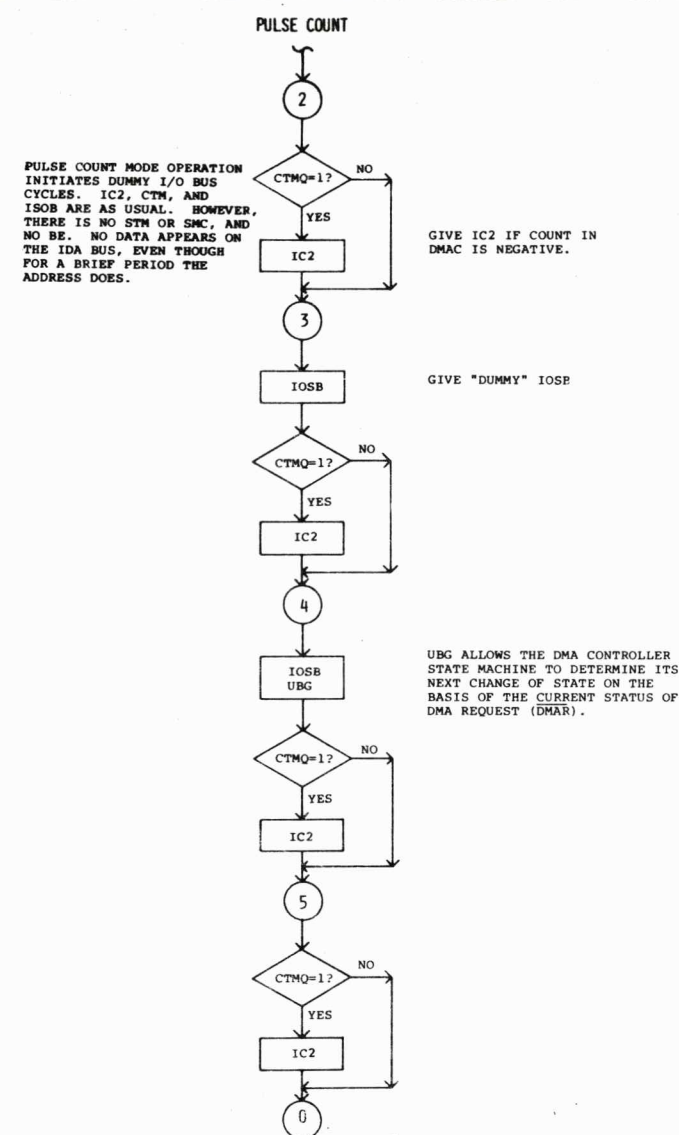


FIG 23-8

SECTION 23 (CONTINUED)

ations that read from memory and write to the peripheral. The first thing that is done is to do the read memory cycle. Recall that the previous state actually began this cycle by sending the address. All that is necessary is to continue sending the address and actually issue STM. The data will be put into the 0 register.

The remainder of the activity in this segment involves a write I/O Bus Cycle to the peripheral. The BIB's connecting the IDA Bus to the IOD Bus are enabled. The IDA Bus is driven with the data in 0. IOSB is given, as is IC2 if the count in DMAC is negative. All of this activity is done for several states. During the

I/O Bus Cycle UBG is issued. This causes the DMA Controller to cycle.

In the event of a DMA operation that reads from a peripheral and writes into memory, the associated memory cycle must be made into a write memory cycle. In addition, the direction of the buffers connecting the IOD Bus and the IDA Bus must be changed. Once the memory cycle is started an I/O Bus Cycle is initiated. The peripheral's response to the I/O Bus Cycle is actually used as the source of the data to be sent to the memory. No intermediate storage is involved. The remaining control signals are as previously described for the other DMA operation.

In both types of DMA operations the IOC acts as the originator of the associated memory cycle. In both cases the memory is the source of Memory Complete. This is in accordance with the memory traffic protocol, which stipulates that the originator supplies STM and the respondent supplies Memory Complete, regardless of whether the cycle is a read or write memory cycle.

Figure 23-8 illustrates the segment of the Bus Controller's ASM chart that deals with Pulse Count Mode operations. The purpose of this segment is to a "dummy" DMA cycle. Since no data is transferred, and no actual memory cycle takes place, the notion of read or write does not apply. The potential memory cycle that was initiated in state one is allowed to lapse without an actual STM being issued. Also, no Buffer Enable is issued nor is Data In issued. In fact, all that is actually issued is an IOSB, and an IC2 if the count in DMAC has gone negative. During the middle of the dummy I/O Bus Cycle UBG is issued to cause the DMA Controller to cycle.

PULSE COUNT

DMA

WRITE TO I VIA ERA

INTERRUPT POLL

CONVENTIONS USED IN THE WAVE FORMS

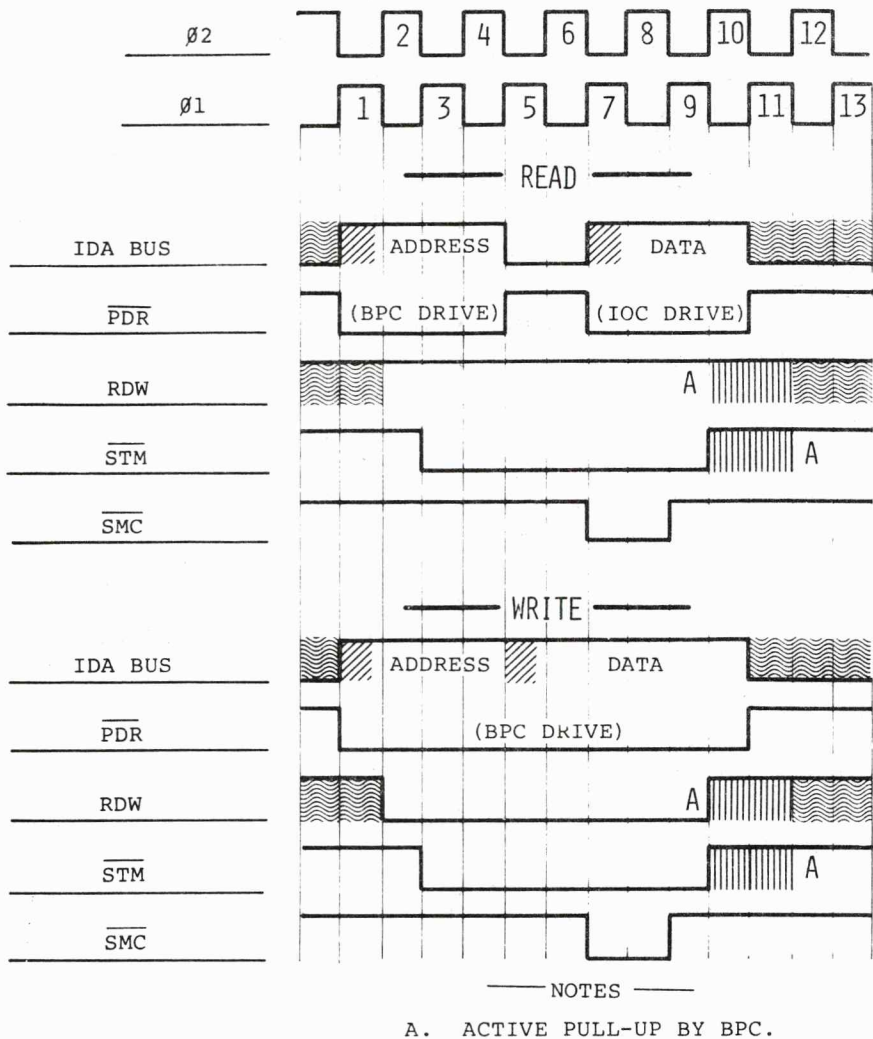


FIG 24-2

CONDENSED RESPONSE TO MEMORY CYCLES
REFERENCING AN INTERNAL IOC REGISTER

1. OR TRANSITION UP OR TRANSITION DOWN CAN OCCUR ANYTIME WITHIN THE INDICATED INTERVAL. USED TO INDICATE TIME-WINDOWS WITHIN WHICH EXTERNALLY ORIGINATED EVENTS CAN HAPPEN. REPRESENTS IDEALIZED LOGICAL ACTIVITY; RISE TIMES AND DELAYS ARE TAKEN INTO CONSIDERATION ONLY IN A GENERAL WAY.
2. OR REPRESENTS THE SET-UP TIME OF A SIGNAL BEING DRIVEN.
3. REPRESENTS A LINE THAT IS EITHER UNDEFINED OR A DON'T CARE.
4. REPRESENTS A LINE THAT IS ACTIVELY PULLED-UP.
5. CAPITAL LETTERS FROM THE START OF THE ALPHABET REPRESENT EXPLANATORY NOTES.
6. NUMERALS IN THE 02-01 WAVEFORMS ARE STRICTLY FOR REFERENCE WITHIN THAT PARTICULAR SET OF WAVEFORMS, AND HAVE NO SIGNIFICANCE OUTSIDE THAT SET.
7. DOTTED LINES INDICATE ZERO OR MORE COMPLETE STATE TIMES THAT OCCUR AS A FUNCTION OF SOME EXTERNAL CONDITION (SUCH AS WAITING FOR MEMORY COMPLETE).
8. IN GENERAL, THE WAVEFORMS ARE QUITE IDEALIZED. THEY EXPLAIN THE LOGICAL RELATIONSHIPS BETWEEN SIGNALS, BUT ACTUAL DELAYS, RISE TIMES, SIGNAL LEVELS AND THRESHOLDS ARE NOT INDICATED.
9. DENOTES THAT A SIGNAL OR NOTE APPLIES TO THE 15-BIT VERSION ONLY. DELETE THE SIGNAL OR NOTE FOR THE 16-BIT VERSION.
10. DENOTES THAT A SIGNAL OR NOTE APPLIES TO THE 16-BIT VERSION ONLY. DELETE THE SIGNAL OR NOTE FOR THE 15-BIT VERSION.

FIG 24-1

REPONSE TO A READ MEMORY CYCLE
REFERENCING AN INTERNAL IOC REGISTER

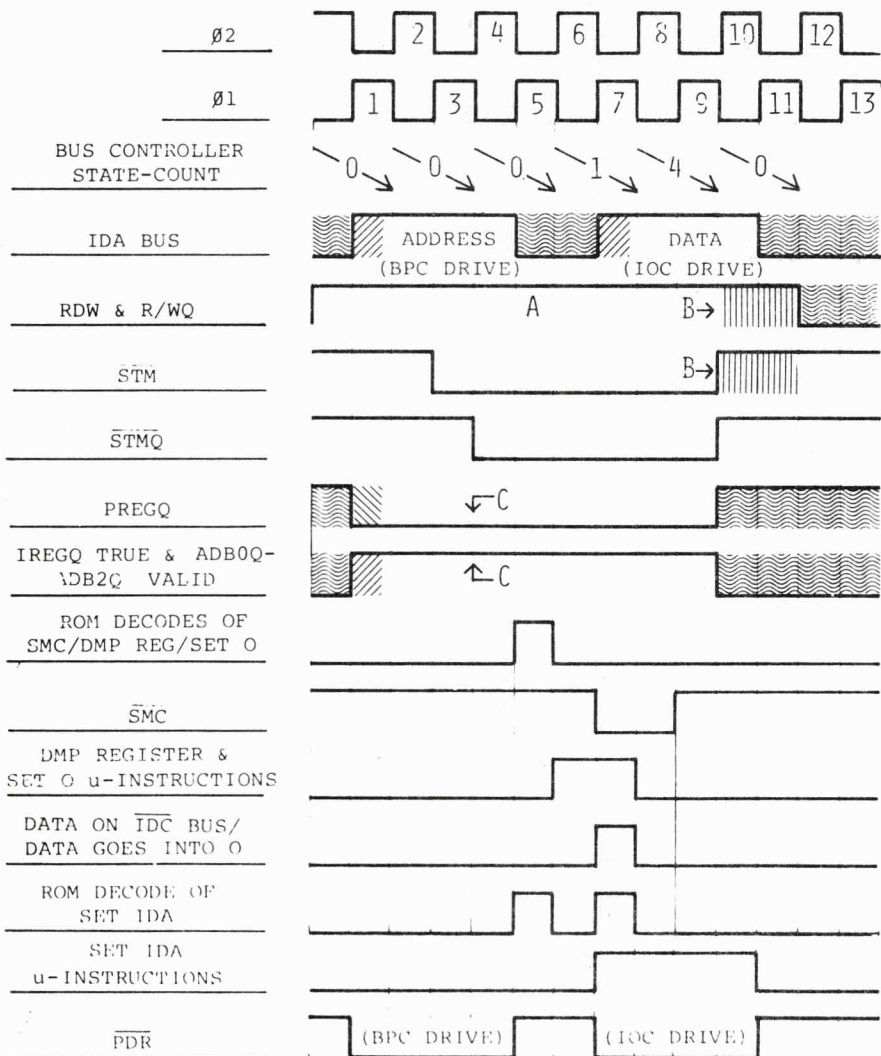
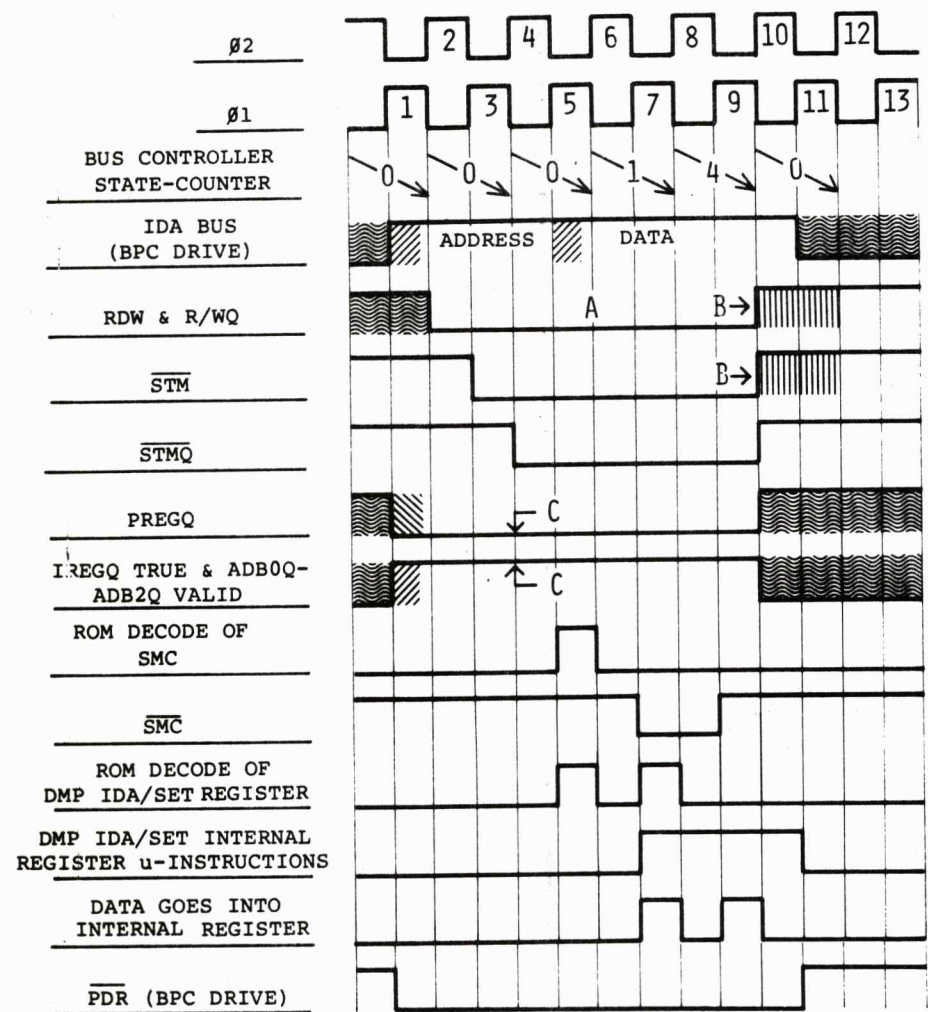


FIG 24-3

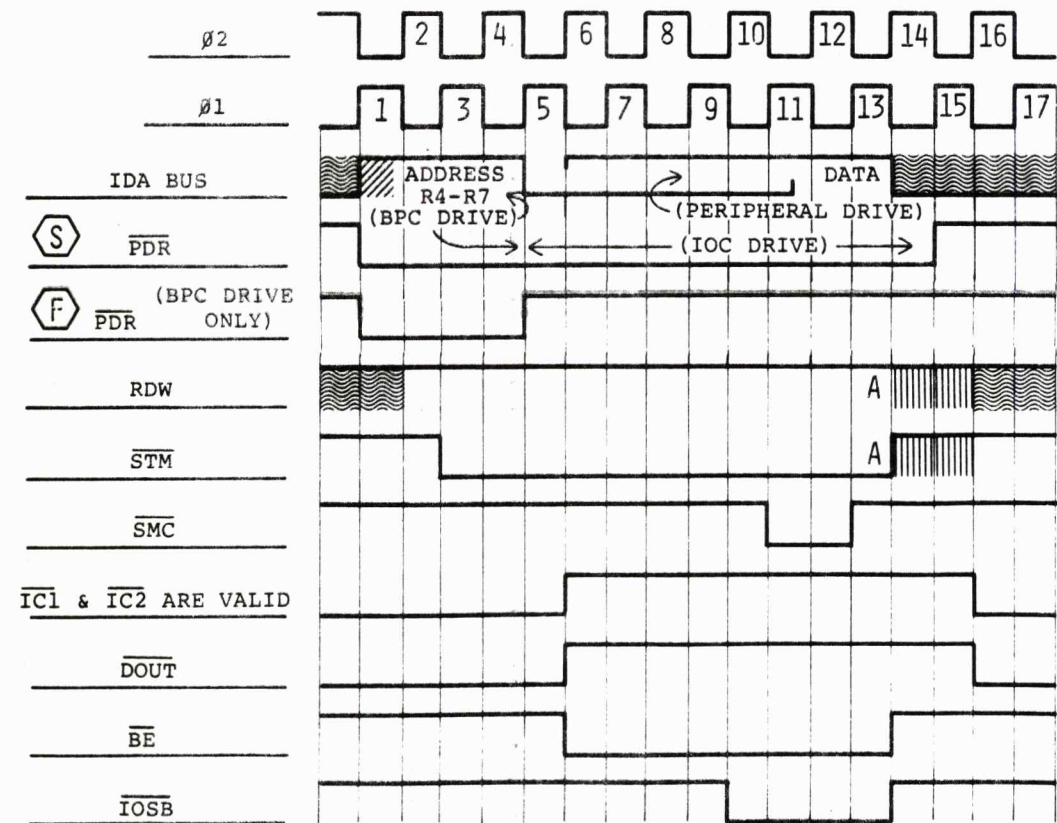
RESPONSE TO A WRITE MEMORY CYCLE
REFERENCING AN INTERNAL IOC REGISTER



- NOTES
- A. ASSUME THE BPC IS THE ORIGINATOR OF THE MEMORY CYCLE. THE IOC LOOKS AT R/WQ DURING CLOCKTIMES 4 AND 6 ONLY.
 - B. ACTIVE PULL-UP ON RDW AND \overline{STM} BY THE BPC, (FOR RDW ONLY -- NOT FOR R/WQ).
 - C. LATCHED BY SAQL (FIRST ø2 FOLLOWING STM). REMAINS LATCHED UNTIL END OF STM.

FIG 24-4

CONDENSED READ I/O BUS CYCLE



- NOTES
- A. ACTIVE PULL-UP BY BPC.

FIG 24-5

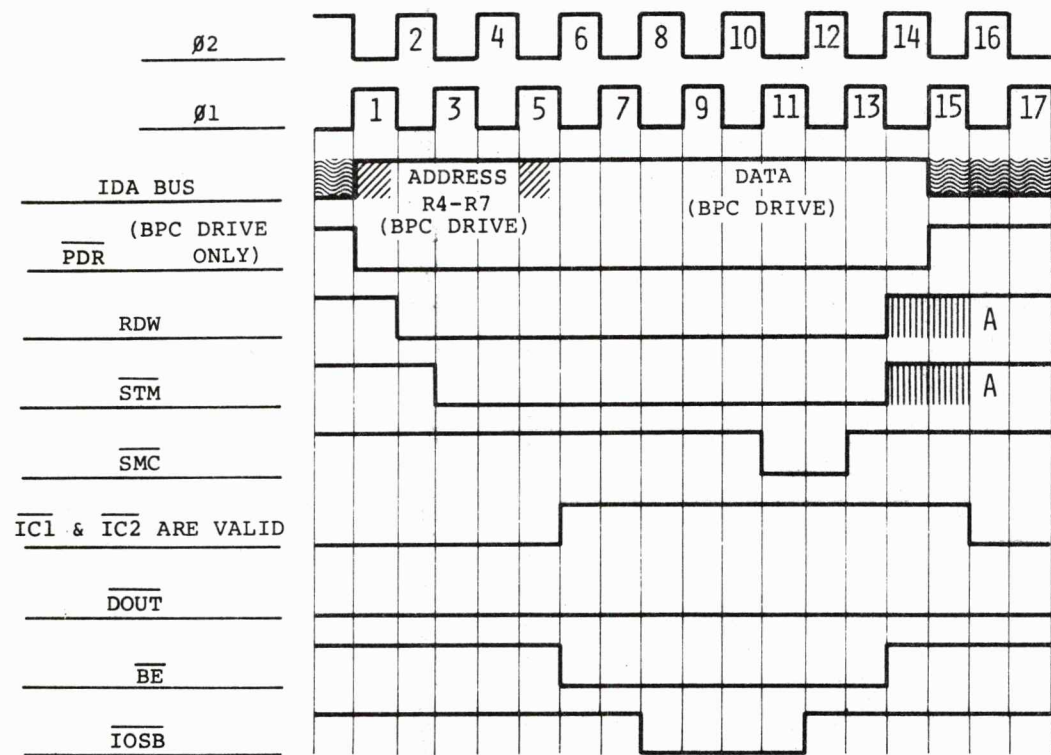
WAVEFORM CONVENTIONS
CONDENSED MEMORY CYCLES
FOR AN INTERNAL IOC REG.

READ
INTERNAL
IOC
REGISTER

WRITE TO AN
INTERNAL
IOC REGISTER

CONDENSED READ
I/O BUS CYCLE

CONDENSED WRITE I/O BUS CYCLE

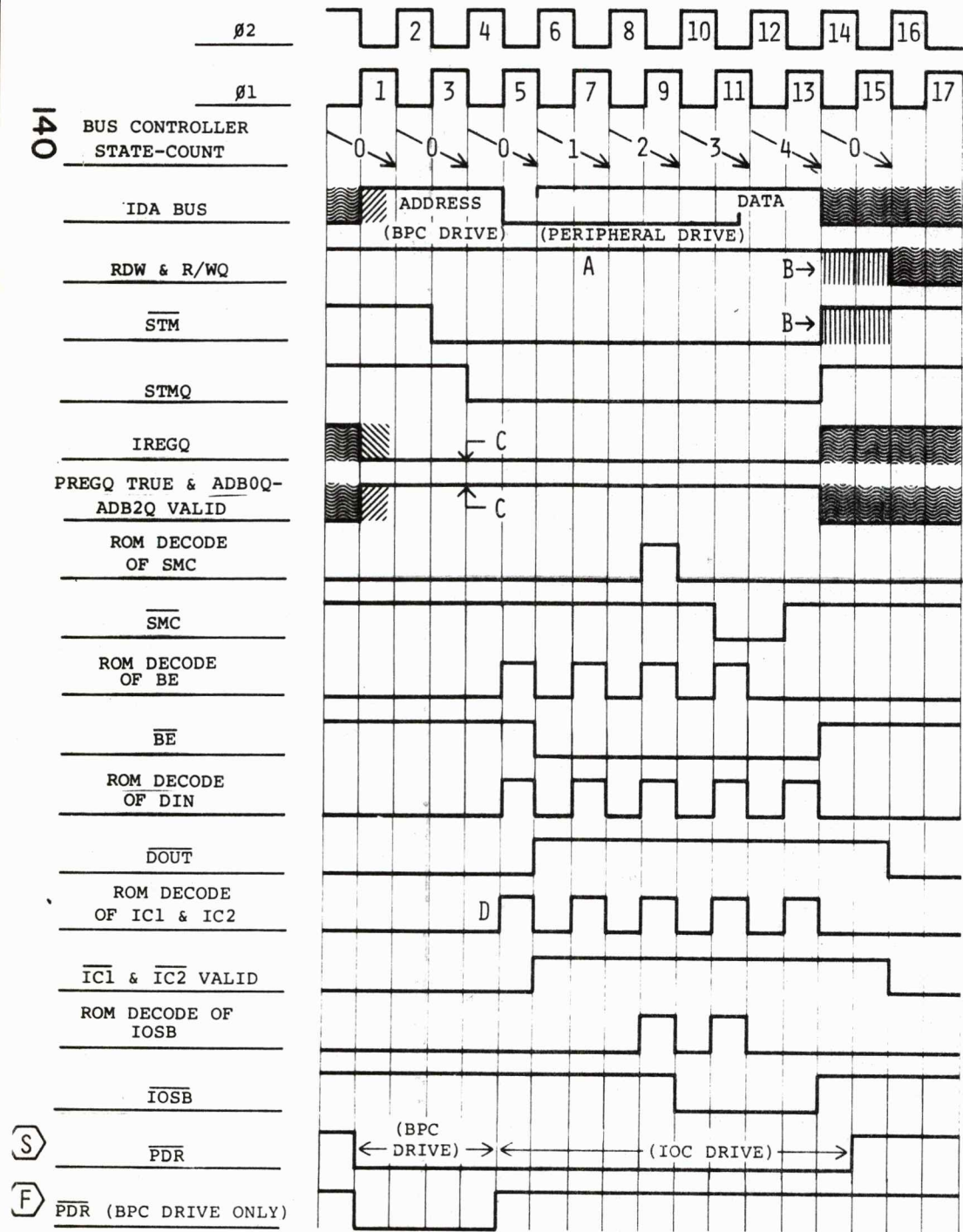


NOTES

A. ACTIVE PULL-UP BY BPC.

FIG 24-6

READ I/O BUS CYCLE



NOTES

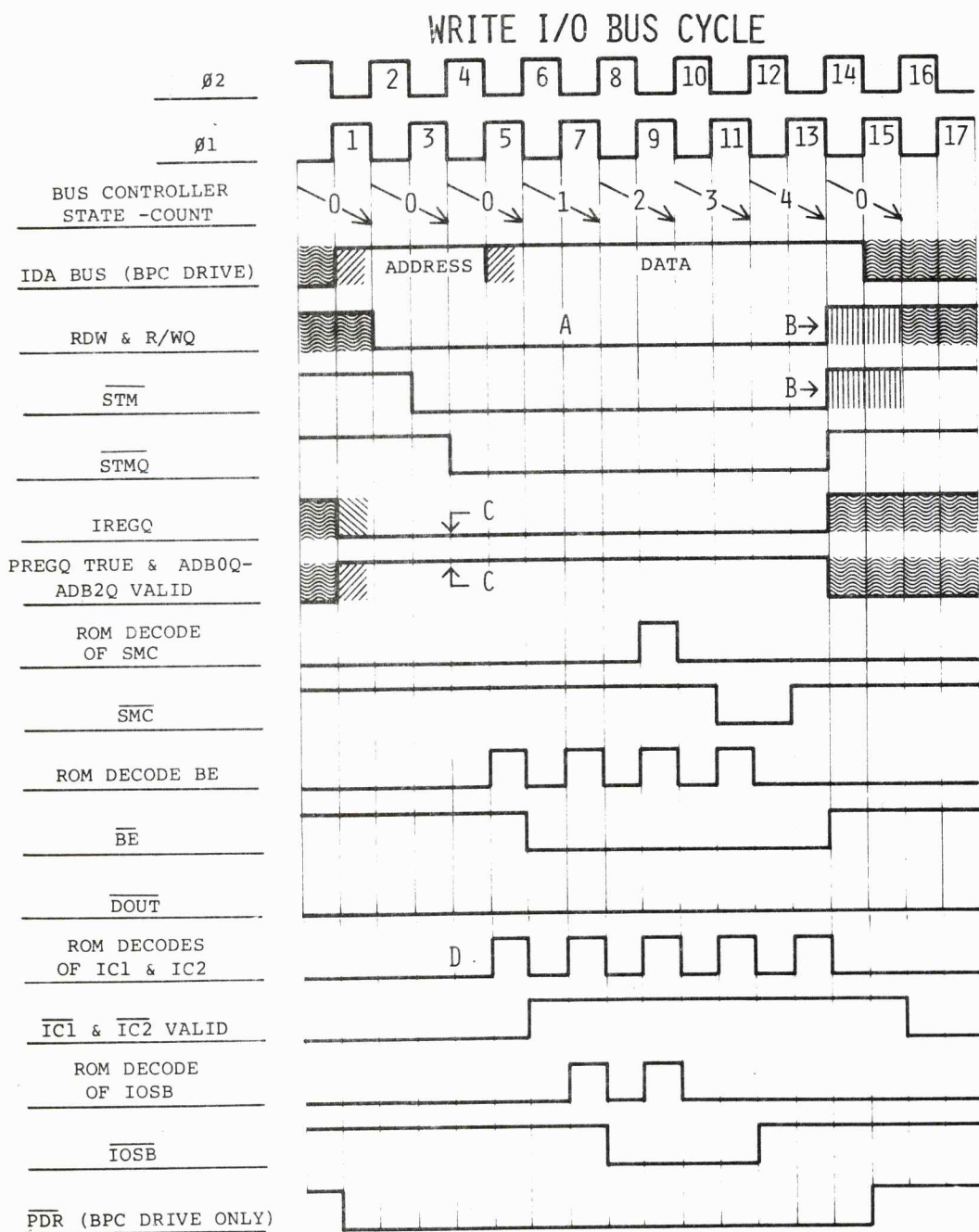
A. ASSUMES THE BPC IS THE ORIGINATOR OF THE MEMORY CYCLE. THE IOC LOOKS AT R/WQ DURING CLOCKTIMES 4,6,8,10&12 ONLY.

B. ACTIVE PULL-UP ON RDW AND STM BY THE BPC, (FOR RDW ONLY -- NOT FOR R/WQ).

C. LATCHED BY SAQL (FIRST ø2 FOLLOWING STM). REMAINS LATCHED UNTIL END OF STM.

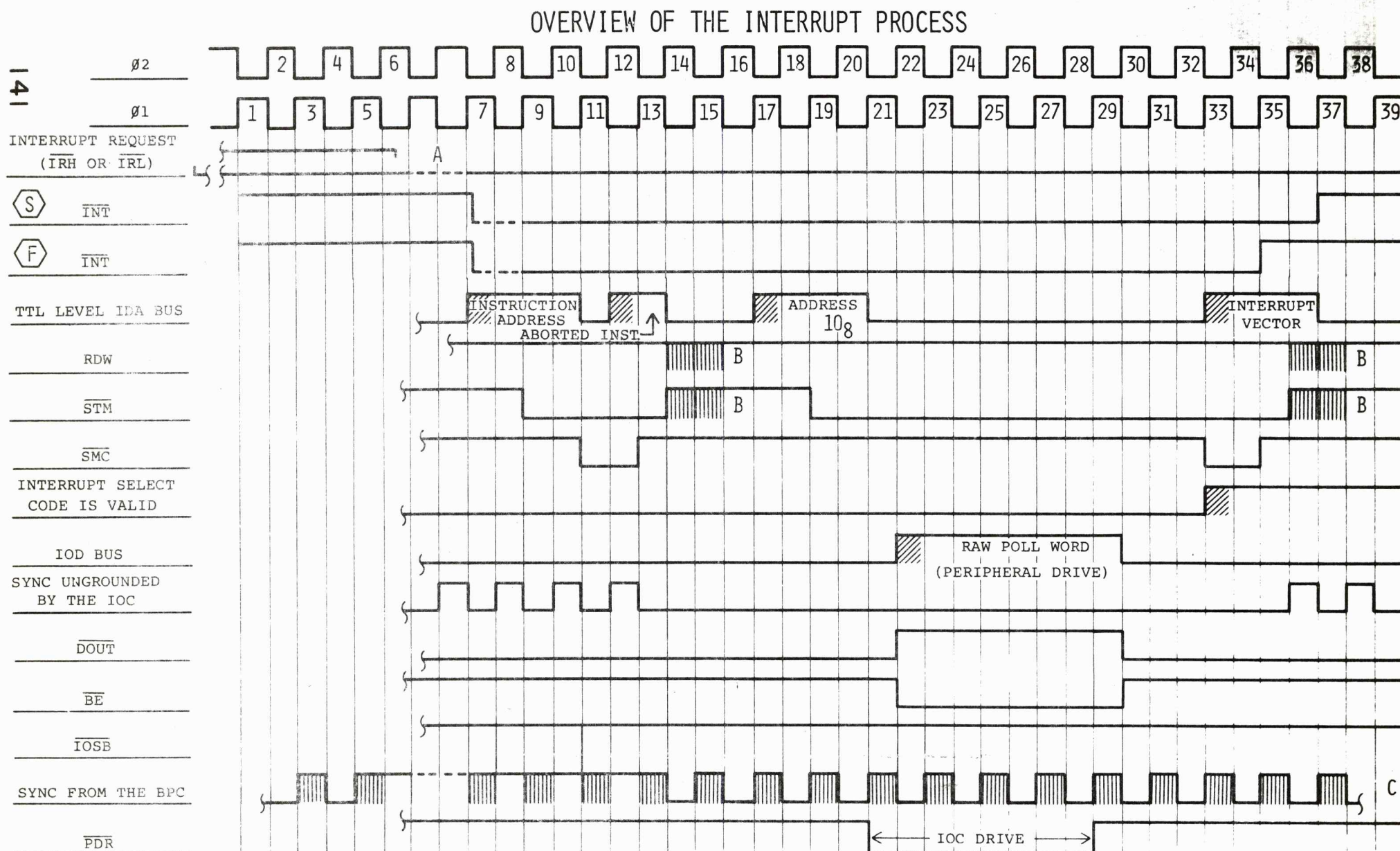
D. WHICH ONES ARE DECODED DEPENDS UPON WHICH OF R4-R7 WAS REFERENCED.

FIG 24-7



- NOTES
- ASSUMES THE BPC IS THE ORIGINATOR OF THE MEMORY CYCLE. THE IOC LOOKS AT R/WQ DURING CLOCKTIMES 4,6,8,10&12 ONLY.
 - ACTIVE PULL-UP ON RDW AND STM BY THE BPC, (FOR RDW ONLY -- NOT FOR R/WQ).
 - LATCHED BY SAQL (FIRST ø2 FOLLOWING STM). REMAINS LATCHED UNTIL END OF STM.
 - WHICH ONES ARE DECODED DEPENDS UPON WHICH OF R4-R7 WAS REFERENCED.

FIG 24-8



- NOTES
- DOTTED LINE REPRESENTS VARIABLE LENGTH DELAY UNTIL AN INTERRUPT REQUEST CAN BE GRANTED BY THE IOC.
 - ACTIVE PULL-UP BY THE BPC.

- EARLIEST POSSIBLE UNGROUNDING OF SYNC DURING ø2 BY THE BPC IS DURING CLOCK TIME 62; ASSUMING THAT BIT 15 OF THE IV REGISTER IS A ONE, THAT THE MEMORY CYCLE TO STORE THE NEWEST VALUE IN THE RETURN STACK IS A MINIMUM MEMORY CYCLE, AND THAT THERE ARE NO INTERVENING BUS REQUESTS.

FIG 24-9

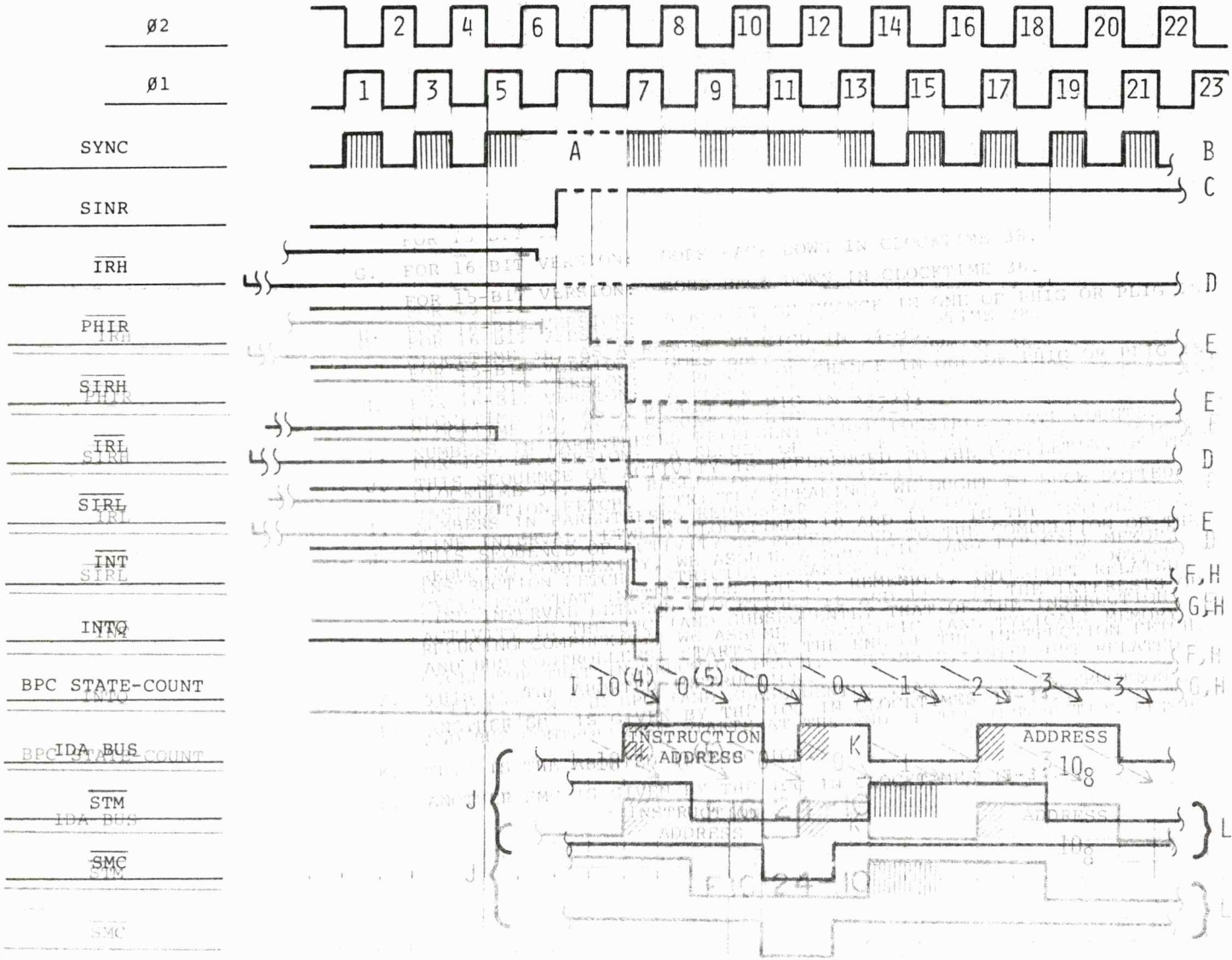
CONDENSED WRITE
I/O BUS CYCLE

READ I/O
BUS CYCLE

WRITE I/O
BUS CYCLE

INTERRUPT PROCESS
OVERVIEW

GENERATION BY THE IOC OF INT FROM AN INTERRUPT REQUEST: EFFECT OF INT UPON THE BPC

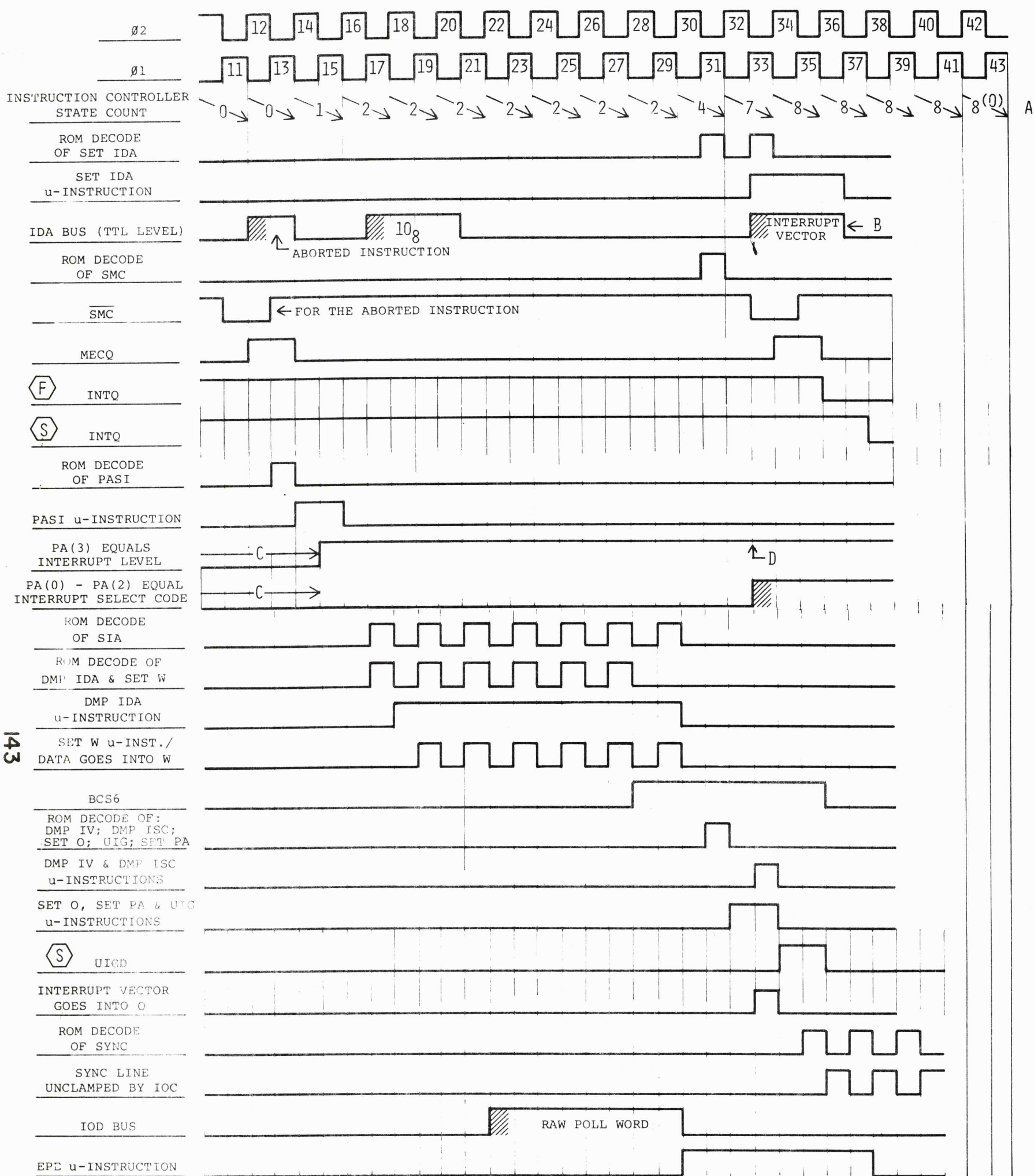


NOTES

- A. THE DOTTED LINE REPRESENTS A VARIABLE NUMBER OF STATE-TIMES. THIS COMES ABOUT BECAUSE THE BPC ALLOWS SYNC AT VARYING INTERVALS BEFORE REACHING ITS INSTRUCTION FETCH SEQUENCE. THE OTHER CHIPS IN THE SYSTEM ALSO HAVE VARIABILITY IN THE MANNER IN WHICH THEY ALLOW SYNC.
- B. WHEN SYNC RESUMES DEPENDS UPON THE NUMBER OF INDIRECT ADDRESS ENCOUNTERED WHILE USING THE INTERRUPT TABLE. CLOCKTIME 40 WOULD BE THE EARLIEST SYNC COULD AGAIN GO HIGH; THAT WOULD BE FOR ONE LEVEL OF INDIRECT.
- C. DEPENDS UPON SYNC.
- D. HELD UNTIL THERE IS AN IOSB WITH THE PROPER SELECT CODE FOR THE INTERRUPTING PERIPHERAL (OCCURS DURING THE INTERRUPT SERVICE ROUTING--NO IOSB OCCURS DURING THE INTERRUPT POLL).
- E. FOLLOWS THE INTERRUPT REQUESTS.
- F. FOR 16-BIT VERSION: COMES BACK UP IN CLOCKTIME 37.
FOR 15-BIT VERSION: COMES BACK UP IN CLOCKTIME 35.
- G. FOR 16-BIT VERSION: GOES BACK DOWN IN CLOCKTIME 38.
FOR 15-BIT VERSION: GOES BACK DOWN IN CLOCKTIME 36.
- H. FOR 16-BIT VERSION: A RESULT OF CHANGE IN ONE OF PHIG OR PLIG IN CLOCKTIME 36, AS A RESULT OF UIGD IN 34-35.
FOR 15-BIT VERSION: A RESULT OF CHANGE IN ONE OF PHIG OR PLIG IN CLOCKTIME 34, AS A RESULT OF UIG IN 32-33.
- I. NUMBERS IN PARENTHESES REPRESENT OTHER POSSIBLE STATE COUNTS.
- J. THIS SEQUENCE OF ACTIVITY IS REFERENCED TO THE COMPLETION OF THE INSTRUCTION FETCH. STRICTLY SPEAKING, WE OUGHT TO SHOW DOTTED LINE INTERVAL BETWEEN CLOCKTIMES 10 AND 11. (IN THE INTERESTS OF REDUCING COMPLEXITY, WE ASSUME A SPECIFIC (AND TYPICAL) MEMORY CYCLE FOR THAT INSTRUCTION FETCH. REMEMBER, INTERRUPT RELATED ACTIVITY IN THE BPC (AND SUBSEQUENTLY THAT OF THE INSTRUCTION AND BUS CONTROLLERS) STARTS AT THE END OF THE INSTRUCTION FETCH.
- K. THIS IS THE ABORTED INSTRUCTION.
- L. ANOTHER SMC IS GIVEN BY THE IOC IN CLOCKTIMES 33-34.

FIG 24-10

GENERATION OF THE INTERRUPT VECTOR BY THE INSTRUCTION CONTROLLER



143

NOTES

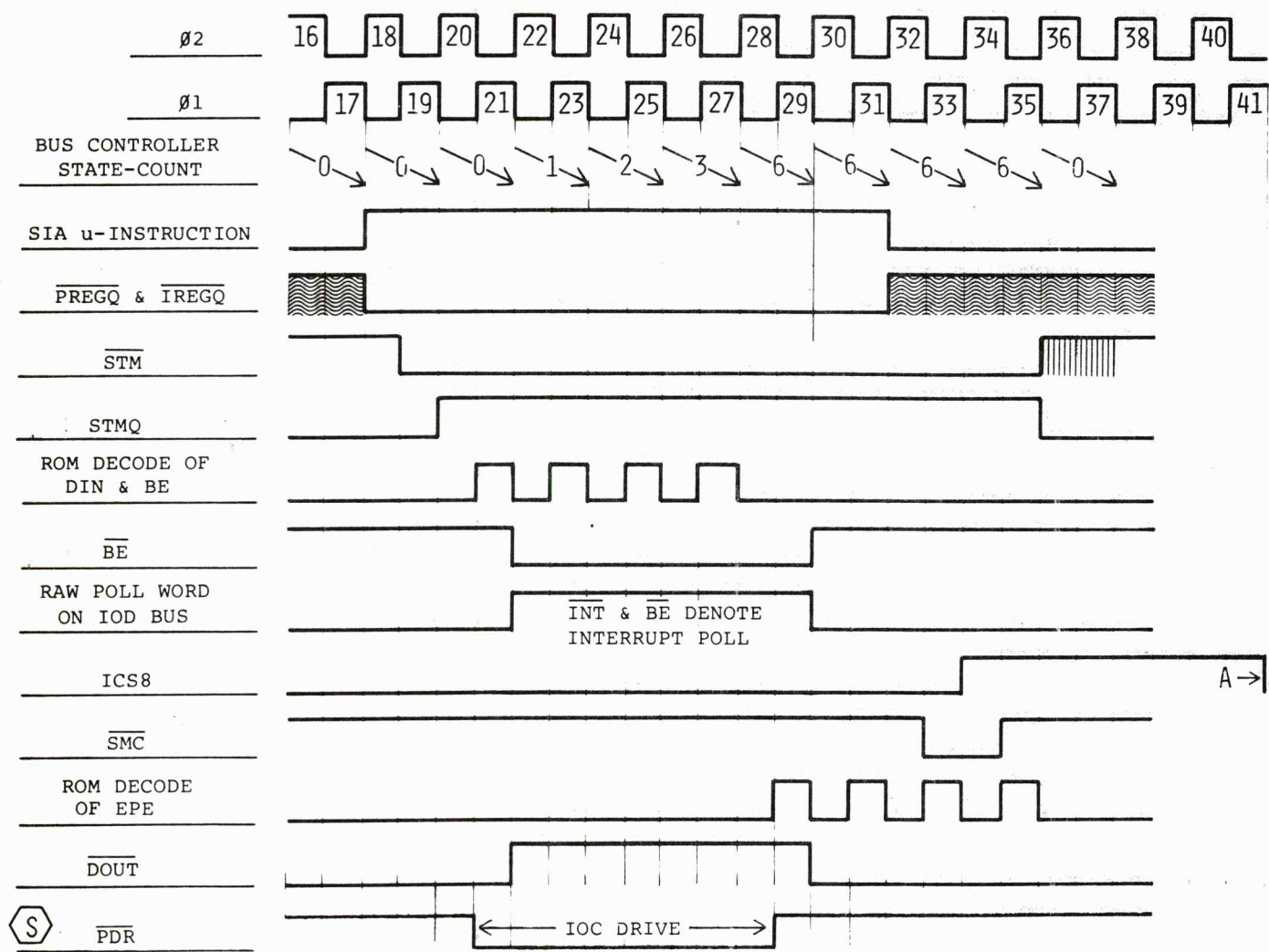
- A. THIS STATE CHANGE AWAITS SYNC, WHICH IS NOT GIVEN BY THE BPC (JSM SEGMENT OF ASM CHART) UNTIL THE JSM-INDIRECT IS COMPLETED. ASSUMING MINIMUM LENGTH MEMORY CYCLES, THE FIRST STATE 0 WOULD BE CLOCK-TIMES 64-65.
- S B. THE WORD IN THE INTERRUPT TABLE (ACCESSED BY DOING A JSM 10₈, 1) MUST BE THE ADDRESS OF THE DESTINATION AND NOT ANOTHER INDIRECT ADDRESS.
- F B. ASSUMING THAT THE WORD IN THE INTERRUPT TABLE (ACCESSED BY DOING A JSM 10₈, 1) IS THE ADDRESS OF THE DESTINATION AND NOT ANOTHER INDIRECT ADDRESS.
- C. PREVIOUS SELECT CODE.
- D. PA(3) IS SET AGAIN BY THE HIGH-LOW INTERRUPT VECTOR GENERATOR.
- E. EARLIEST THAT SYNC CAN GO HIGH IS IN CLOCK TIME 62.

FIG 24-11

GENERATION OF INT,
EFFECT ON BPC

GENERATION OF THE
INTERRUPT VECTOR

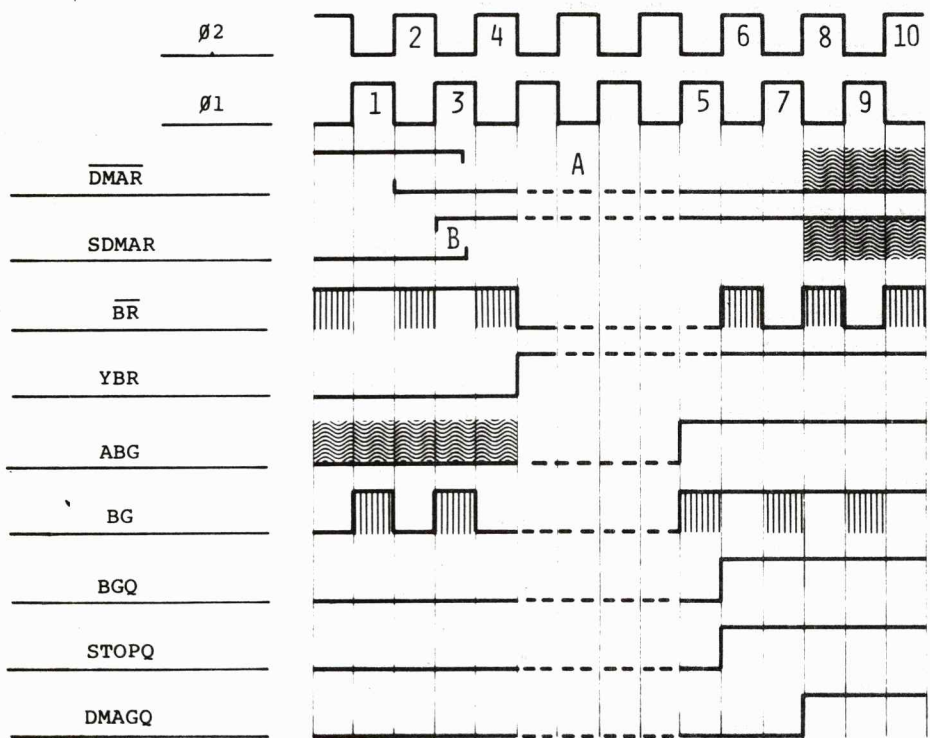
PERFORMANCE OF THE INTERRUPT POLL BY THE BUS CONTROLLER



A. IN 15-BIT VERSION ONLY: EARLIEST POSSIBLE TRANSITION. DEPENDS UPON NUMBER OF LEVELS OF INDIRECT THROUGH INTERRUPT TABLE.

FIG 24-12

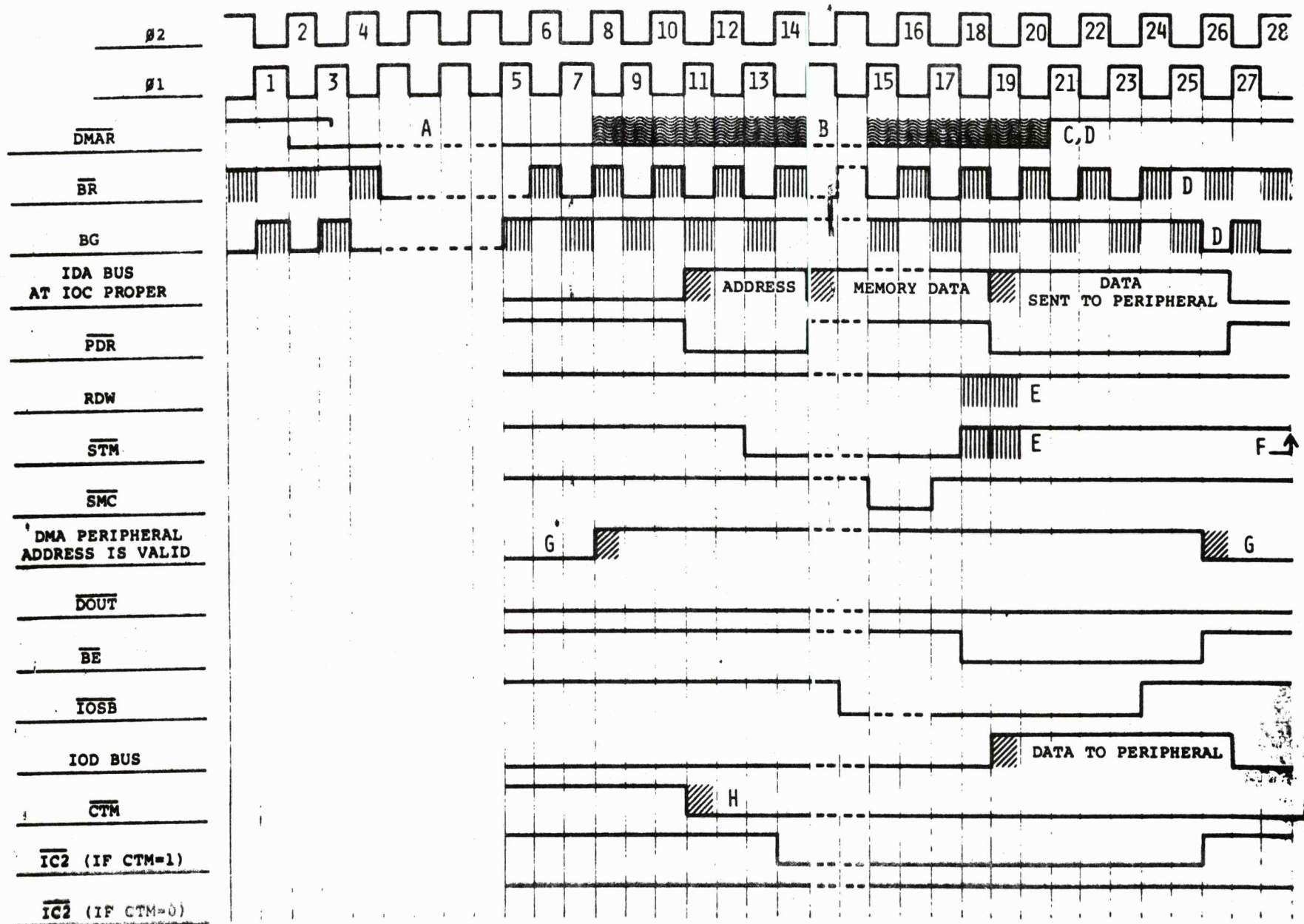
GENERATION OF BUS GRANT AND DMAGQ FROM DMA REQUEST (EITHER DMA OR PCM MODE)



- NOTES
- A. DOTTED LINE REPRESENTS POSSIBLE WAIT UNTIL ABG IS GIVEN BY THE INSTRUCTION CONTROLLER, AND BG IS GIVEN BY THE BPC.
 - B. FOLLOWS DMAR.

FIG 24-13

CONDENSED DMA READ FROM MEMORY

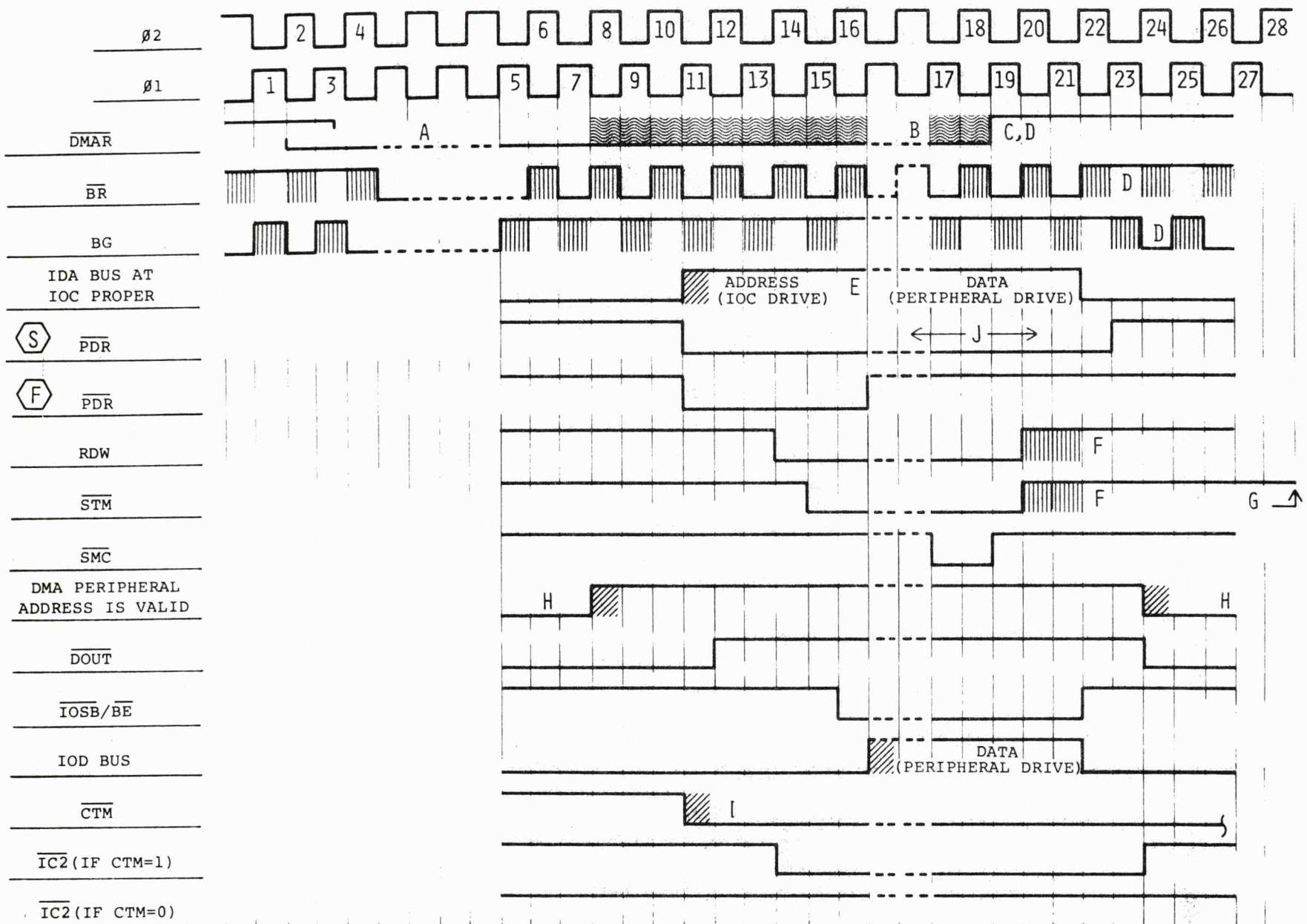


NOTES

- POSSIBLE WAIT UNTIL THE IOC ALLOWS BUS GRANT, AND THE BPC GIVES A BUS GRANT.
- VARIABLE TIMING FOR MEMORY CYCLE-POSSIBLE WAIT FOR SMC.
- AT CLOCK TIME 21 DMA REQUEST MUST REFLECT WHETHER OR NOT THERE IS TO BE A CONSECUTIVELY FOLLOWING DMA CYCLE.
- WAVEFORM SHOWN ASSUMES NO CONSECUTIVELY FOLLOWING DMA CYCLE.
- ACTIVE PULL-UP BY BPC
- EARLIEST TRANSITION IF THERE WERE A CONSECUTIVELY FOLLOWING DMA CYCLE.
- NON-DMA PERIPHERAL ADDRESS IS ON THE BUS.
- ASSUMING THE COUNT IN DMAC GOES NEGATIVE.

FIG 24-14

CONDENSED DMA WRITE INTO MEMORY



NOTES

- POSSIBLE WAIT UNTIL THE IOC ALLOWS BUS GRANT, AND THE BPC GIVES A BUS GRANT.
- VARIABLE TIMING FOR MEMORY CYCLE-POSSIBLE WAIT FOR SMC.
- AT CLOCK TIME 21 DMA REQUEST MUST REFLECT WHETHER OR NOT THERE IS TO BE A CONSECUTIVELY FOLLOWING DMA CYCLE.
- WAVEFORM SHOWN ASSUMES NO CONSECUTIVELY FOLLOWING DMA CYCLE.
- CLOCK TIME 16 IS A CONFLICT: BOTH THE IOC AND THE PERIPHERAL ARE DRIVING THE BUS.
- ACTIVE PULL-UP BY THE BPC.
- EARLIEST TRANSITION IF THERE WERE A CONSECUTIVELY FOLLOWING DMA CYCLE.
- NON-DMA PERIPHERAL ADDRESS IS ON THE BUS.
- ASSUMING THE COUNT IN DMAC GOES NEGATIVE.
- THE 16-BIT IOC'S PDR IS THE "OR" OF THE IOC'S SET IDA AND DATA IN.

FIG 24-15

DMA READ FROM MEMORY

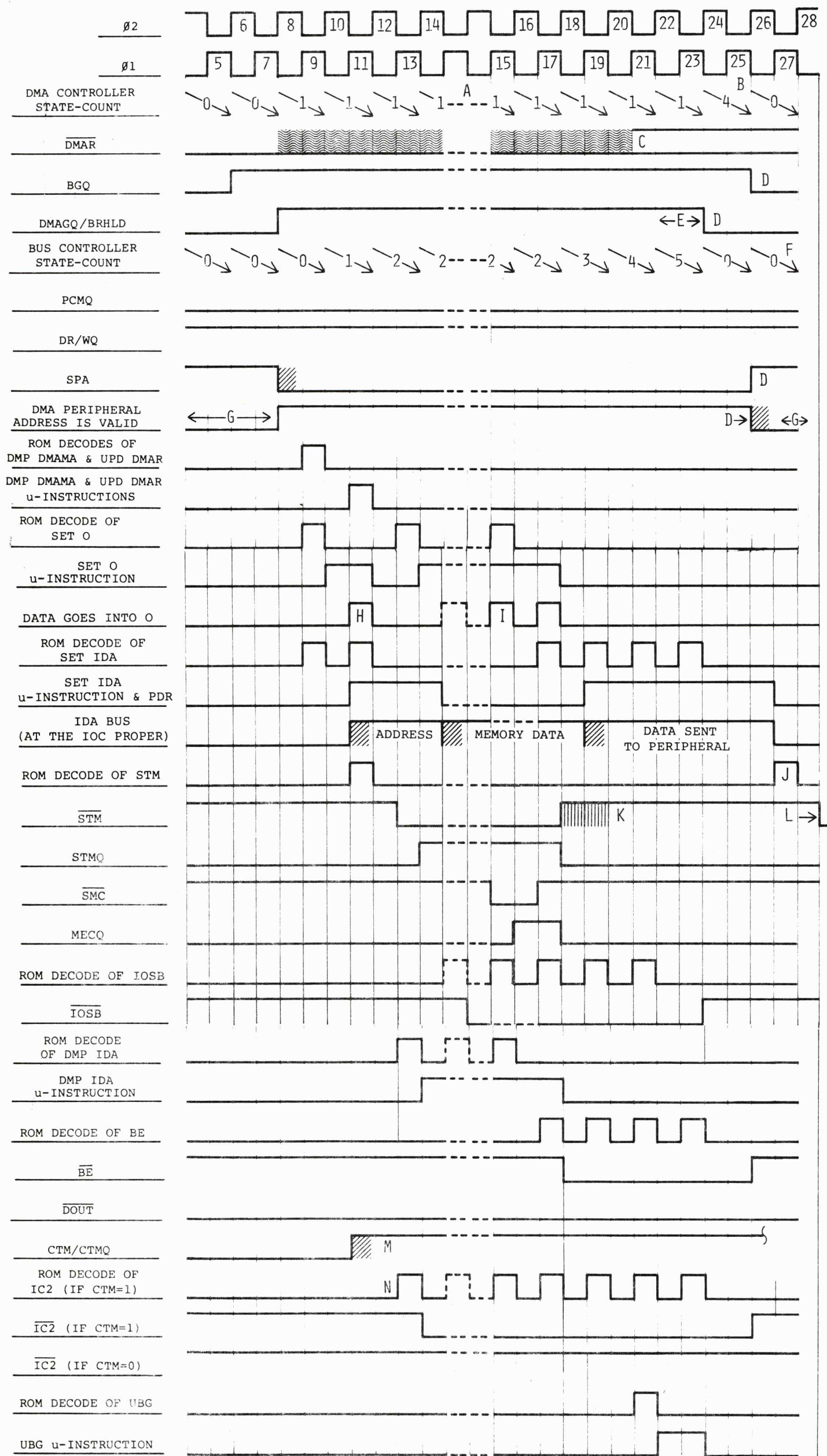


FIG 24-16

CONDENSED DMA WRITE
INTO MEMORY

DMA READ
FROM MEMORY

NOTES

- A. THE DOTTED LINE REPRESENTS A VARIABLE NUMBER OF STATE-TIMES. THIS COMES ABOUT BECAUSE OF POSSIBLE VARIATIONS IN MEMORY CYCLE TIMING.
- B. ASSUMES NO CONSECUTIVELY FOLLOWING DMA CYCLE. HAD THERE BEEN, THIS STATE-COUNT WOULD REMAIN 1 FOR ANOTHER COMPLETE CYCLE.
- C. IT IS AT THIS TIME THAT DMAR MUST REFLECT WHETHER OR NOT THERE IS TO BE A CONSECUTIVELY FOLLOWING DMA CYCLE.
- D. INDICATED TRANSITION ASSUMES THERE IS TO BE NO CONSECUTIVELY FOLLOWING DMA CYCLE.
- E. DMAR AFFECTS THE DMA CONTROLLER STATE COUNT (AND THUS DMAGQ AND BRHLD) .
- F. IF THERE WERE A CONSECUTIVELY FOLLOWING DMA CYCLE THIS STATE-COUNT WOULD BE 1, OTHERWISE THIS STATE-COUNT WOULD REMAIN 0 UNTIL THE NEXT IOC-RELATED MEMORY CYCLE.
- G. NON-DMA PERIPHERAL ADDRESS IS ON THE PERIPHERAL ADDRESS BUS.
- H. DMA MEMORY ADDRESS GOES INTO 0; IT IS SUBSEQUENTLY SENT OUT AS AN ADDRESS.
- I. MEMORY DATA GOES INTO 0; IT IS SUBSEQUENTLY SENT AS DATA TO THE PERIPHERAL..
- J. EARLIEST NEXT ROM DECODE OF STM FOR A CONSECUTIVELY FOLLOWING DMA CYCLE.
- K. ACTIVE PULL-UP BY THE BPC.
- L. EARLIEST NEXT START MEMORY FOR A CONSECUTIVELY FOLLOWING DMA CYCLE.
- M. INDICATED TRANSITION ASSUMES THE COUNT IN DMAC GOES NEGATIVE DURING THE CURRENT DMA CYCLE.
- N. NOT DECODED IF CTM0 IS FALSE.

148

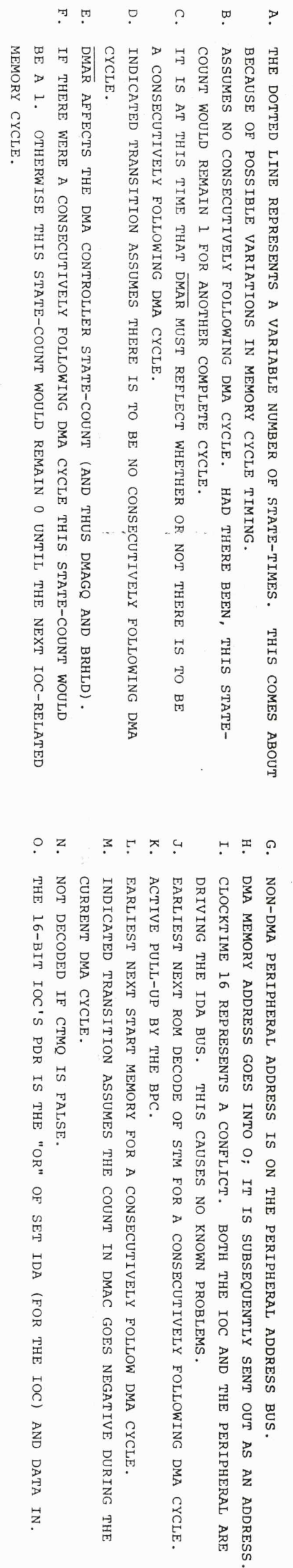
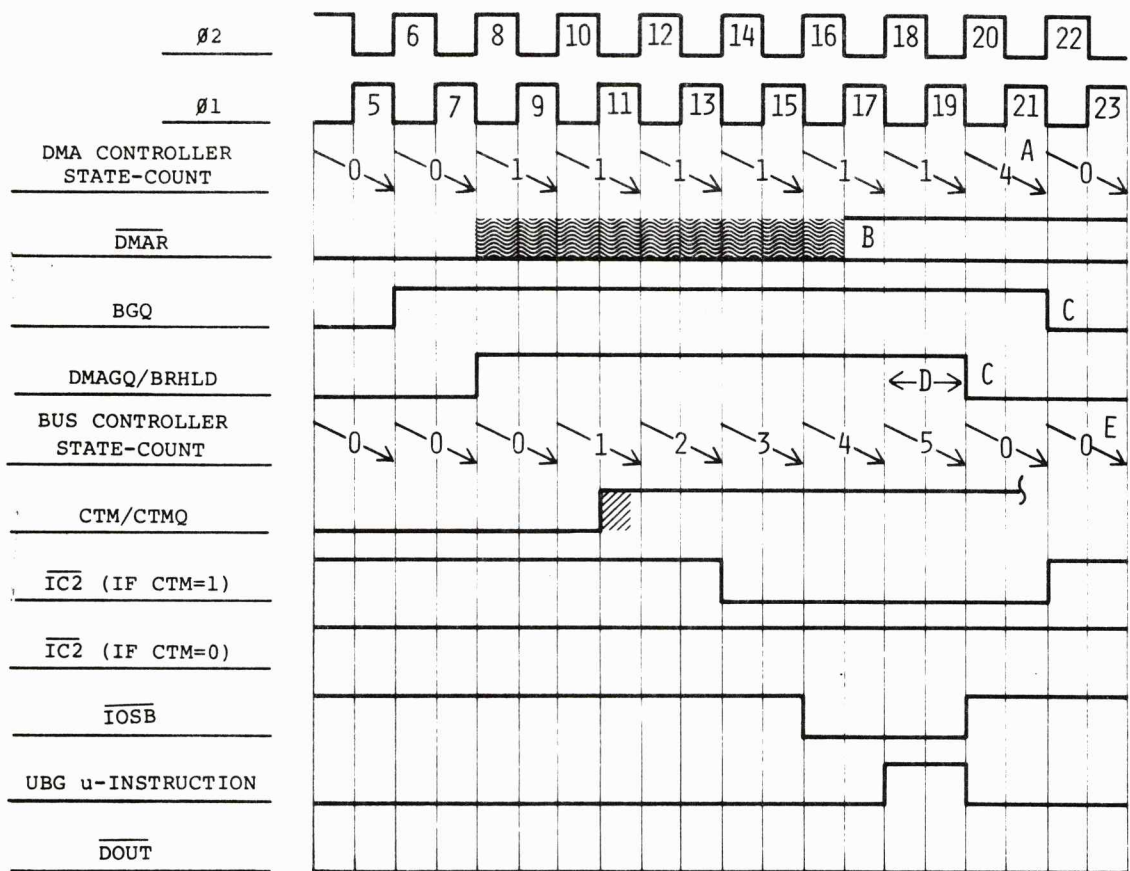


FIG 24-17

PULSE COUNT CYCLE

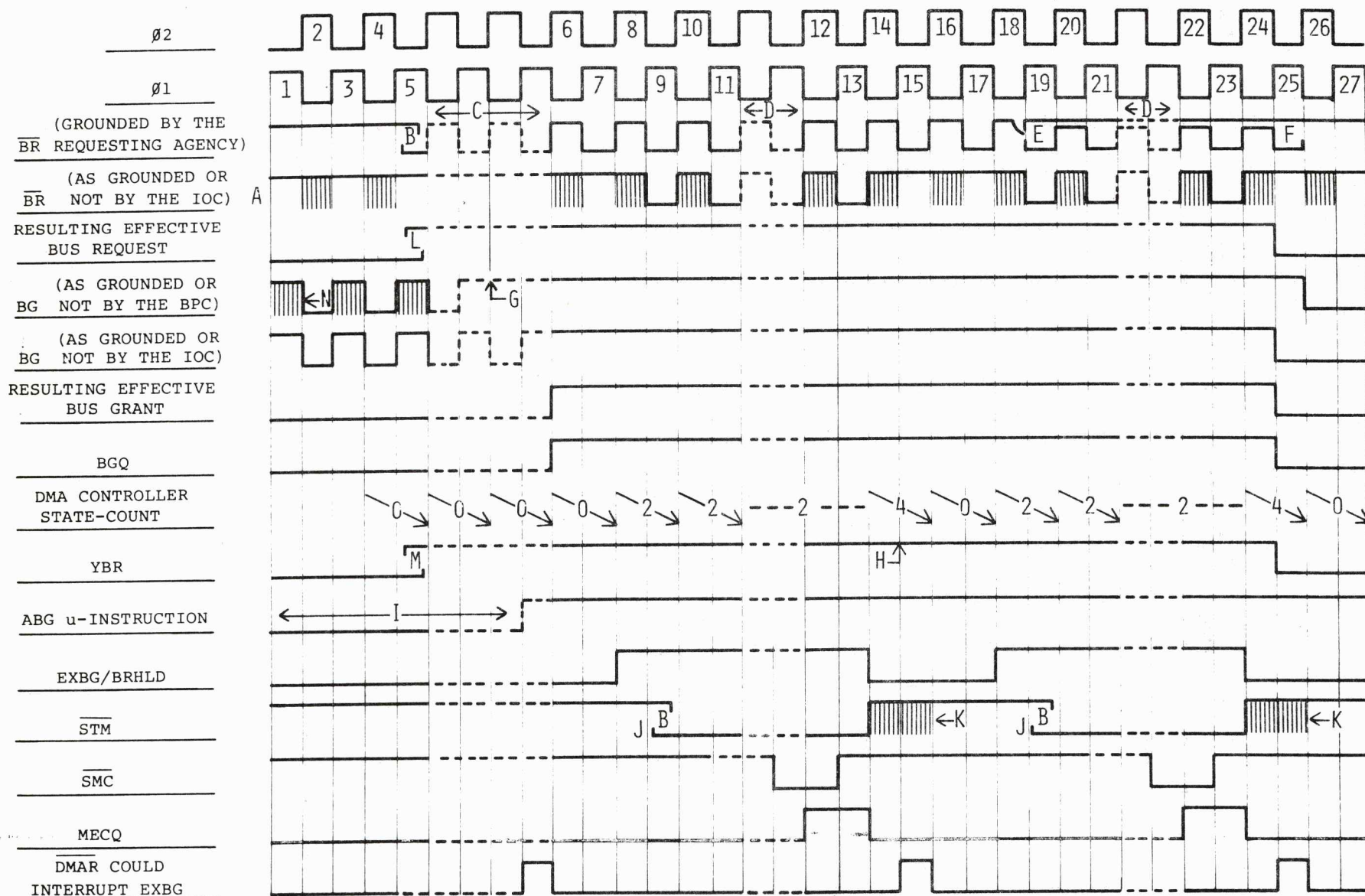


NOTES

- ASSUMES NO CONSECUTIVELY FOLLOWING PULSE COUNT CYCLE. HAD THERE BEEN, THIS STATE-COUNT WOULD REMAIN 1 FOR ANOTHER COMPLETE CYCLE.
- IT IS AT THIS TIME THAT $\overline{\text{DMAR}}$ MUST REFLECT WHETHER OR NOT THERE IS TO BE A CONSECUTIVELY FOLLOWING PULSE COUNT CYCLE.
- INDICATED TRANSITION ASSUMES THERE IS TO BE NO CONSECUTIVELY FOLLOWING PULSE COUNT CYCLE.
- $\overline{\text{DMAR}}$ AFFECTS THE DMA CONTROLLER STATE-COUNT (AND THUS DMAGQ AND BRHLD).
- IF THERE WERE A CONSECUTIVELY FOLLOWING PULSE COUNT CYCLE THIS STATE-COUNT WOULD BE 1. OTHERWISE IT REMAINS A 0 UNTIL THE NEXT IOC-RELATED MEMORY CYCLE.

FIG 24-18

TWO CONSECUTIVE EXTENDED BUS GRANT CYCLES

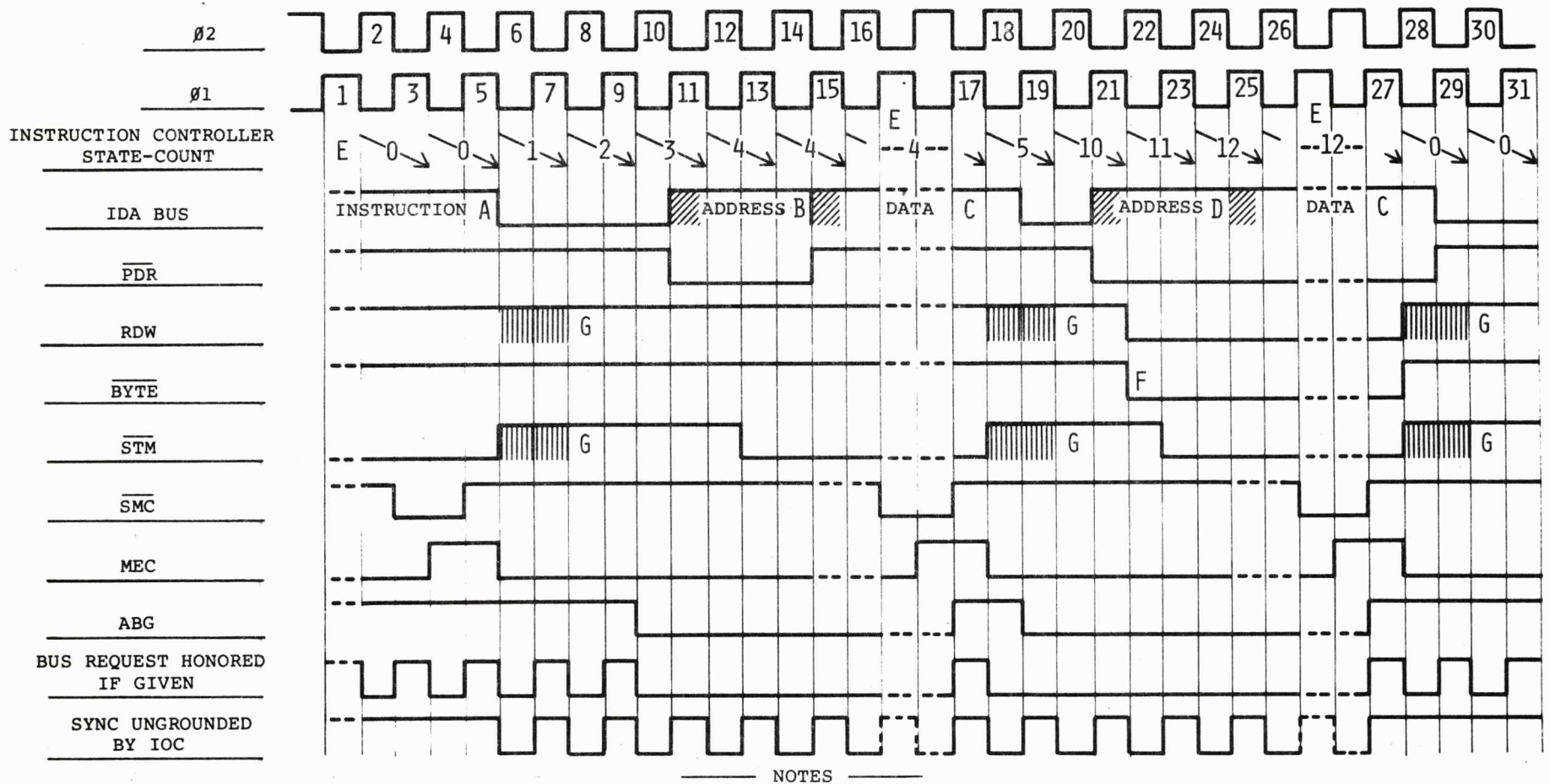


NOTES

- A. THE IOC PRE-CHARGES BUS REQUEST EACH $\phi 2$.
- B. MUST BE WELL GROUNDED SOME MINIMUM TIME BEFORE THE END OF $\phi 1$.
- C. THIS INTERVAL REPRESENTS A WAIT UNTIL THE "AND" OF TWO CONDITIONS IS TRUE. THESE CONDITIONS ARE: (1) THE BPC ALLOWS BUS GRANT, AND (2) ALLOW BUS GRANT (ABG) IS GIVEN. THE WAVE FORMS SHOW THE BPC ALLOWING BUS GRANT PRIOR TO ABG'S BEING GIVEN. THERE ARE OTHER POSSIBILITIES: THE ORDER COULD BE REVERSED, THEY COULD BE CO-INCIDENT, OR, ABG COULD HAVE ALREADY BEEN TRUE FOR A LONG TIME.
- D. REPRESENTS AN INDEFINITE WAIT FOR THE COMPLETION OF THE MEMORY (BUS) CYCLE.
- E. EARLIEST POINT FOR THE AGENCY REQUESTING THE BUS TO CEASE REQUESTING THE BUS, AND STILL INITIATE THE LAST CYCLE.
- F. LATEST POINT FOR THE AGENCY REQUESTING THE BUS TO CEASE REQUESTING THE BUS, WITHOUT INITIATING A SUBSEQUENT CYCLE.
- G. EARLIEST POSSIBLE INSTANCE OF THE BPC'S ALLOWING BUS GRANT IS THE FIRST $\phi 2$ AFTER THE $\phi 1$ WHEN THE BUS IS REQUESTED.
- H. YBR STAYS HIGH HERE BECAUSE $\overline{\text{BUS REQUEST}}$ IS STILL HELD BY THE REQUESTING AGENCY.
- I. ALLOW BUS GRANT (ABG) COULD HAVE BEEN TRUE DURING THE INDEFINITE FUTURE.
- J. START MEMORY (STM) NEEDN'T BE GIVEN RIGHT AWAY, SOONEST STM FOLLOWING EXTENDED BUS GRANT IS SHOWN. THERE REALLY OUGHT TO BE ANOTHER DOTTED LINE SEGMENT BETWEEN CLOCK-TIMES 7 AND 8, AND BETWEEN 17 AND 18.
- K. ACTIVE PULL UP BY THE BPC.
- L. FOLLOWS $\overline{\text{BUS REQUEST}}$ AND GROUNDED BY THE REQUESTING AGENCY.
- M. FOLLOWS THE RESULTING EFFECTIVE BUS REQUEST.
- N. PRE-CHARGED ON $\phi 1$ BY THE BPC.

FIG 24-19

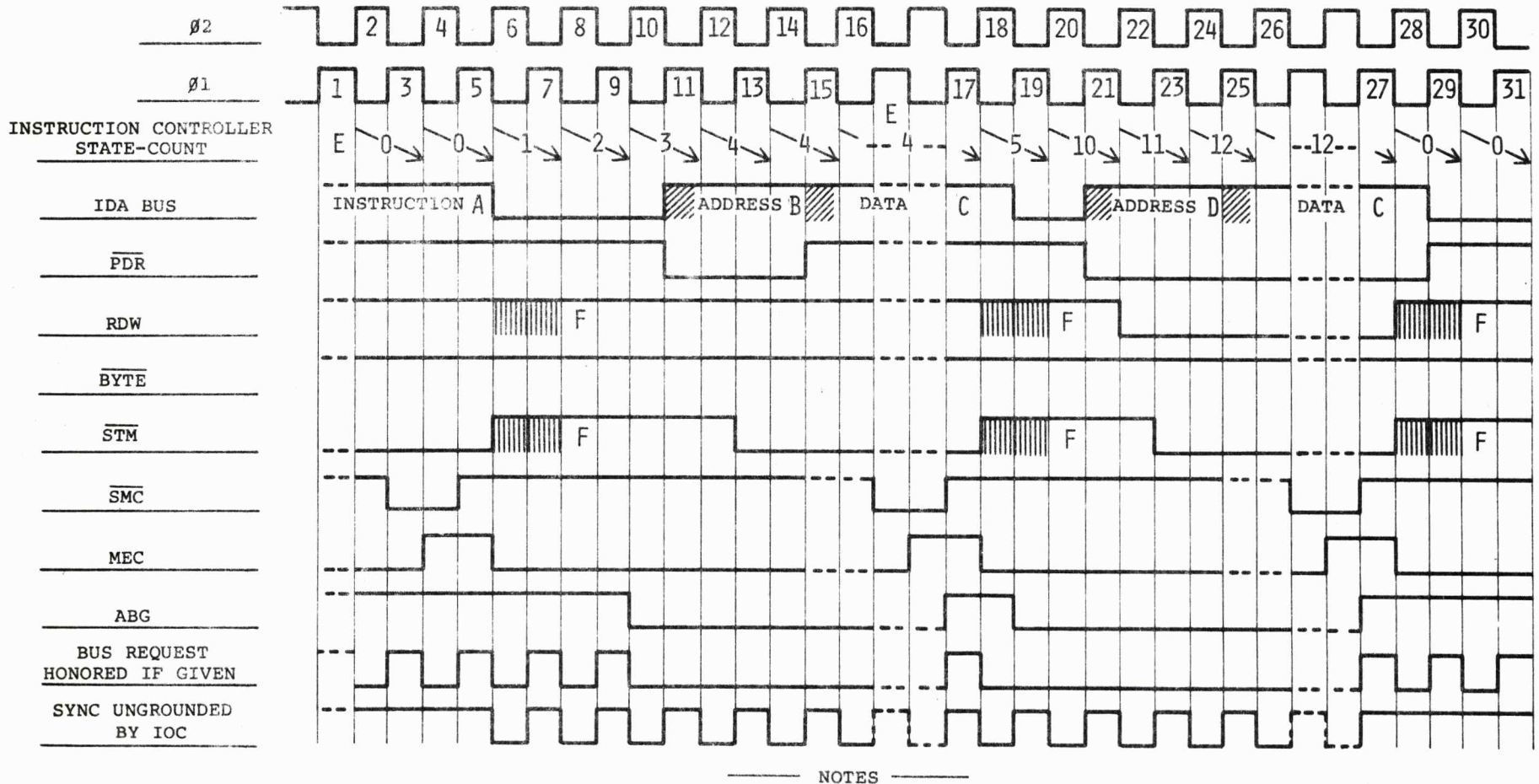
PLACE WORD OR PLACE BYTE



- INSTRUCTION FETCH BY THE BPC.
- ADDRESS OF THE SOURCE REGISTER (0-7) CONTAINING THE DATA TO BE TRANSFERRED.
- THIS IS THE DATA TO BE TRANSFERRED.
- ADDRESS OF THE DESTINATION (STACK LOCATION) OF THE DATA, (CONTENTS OF C OR D REGISTER).
- DOTTED LINES REPRESENT POSSIBLE TIMING VARIATIONS.
- BYTE GOES LOW AS SHOWN ONLY IF THE PLACE INSTRUCTION IS A PLACE-BYTE INSTRUCTION. ALSO, THAT IS THE ONLY TIME THE IOC GROUNDS BYTE.
- ACTIVE PULL-UP BY BPC.

FIG 24-20

WITHDRAW WORD OR WITHDRAW BYTE



- INSTRUCTION FETCH BY THE BPC.
- ADDRESS OF THE SOURCE (STACK LOCATION) OF THE DATA TO BE TRANSFERRED (CONTENTS OF THE COR D REGISTER).
- THIS IS THE DATA TO BE TRANSFERRED.
- ADDRESS OF THE DESTINATION REGISTER (0-7).
- DOTTED LINES REPRESENT POSSIBLE TIMING VARIATIONS
- ACTIVE PULL-UP BY BPC.

FIG 24-21

TWO CONSECUTIVE EXBG

PLACE WORD OR
PLACE BYTE

WITHDRAW WORD OR
WITHDRAW BYTE

CONDENSED TABLE OF THE INSTRUCTION CONTROLLER
AND BUS CONTROLLER ROMS IN THE IOC

MICRO-INSTRUCTIONS			QUALIFIERS																									
STATE-COUNT	OTHER	OUTPUT LINE #	MEMORY & I/O				QUALIFIER						MACHINE				MEMORY					INSTRUCTION				BUS		
			CONTROL				MULTIPLEXER						INSTRUCTION				ADDRESS					STATE-COUNT				STATE-COUNT		
			STMQ	MECQ	STOPQ	DMAGQ	R/WQ OR	DR/WQ	INTQ OR	PCMQ	SYNCQ OR	CTMQ	IB6Q	IB5Q	IB4Q	IB3Q	IREGQ	PREGQ	ADB2Q	ADB1Q	ADB0Q	ISL3Q	ISL2Q	ISL1Q	ISL0Q	BSL2Q	BSL1Q	BSL0Q
	EPCMR	1			0				0			0	1	0	1						0	0	1	0				
NISL0		2		1																	0	0	0	0				
	SYNC	3		0																	0	0	0	0				
	SYNC	4			0				0				0	0	0	0					0	0	1	0				
	SYNC	5											0								1	0	0	0				
	SYNC	6		1					0				1								1	1	0	0				
	ABG	7																				0	0	0				
NISL1	ABG	8																			0	0	0	1				
	ABG	9											0	0							0	0	1	0				
	ABG	10											0								0	1	0	0				
	ABG	11		1					0				1									1	0	0				
	ABG	12			X																							
NBSL2	UBG	13				1	1	1																	1	0	0	
NBSL2	UBG	14		1		1	0	1																	1	0	0	
NBSL2	UBG	15				1		0																	1	0	0	
	UINTR	16			0			0					0	0	1						0	0	1	0				
	EINTR	17			0			0					0	0	1	0					0	0	1	0				
NISL2		18						1													0	0	1	0	1			
	UDMAR	19			0			0					0	1							0	0	1	0				
NISL3		20						1													0	1	1	1				
	EDMAR	21			0			0					0	1	0						0	0	1	0				
NISL3		22								0			0								1	0	0	0				
NISL2	UIG	23						1					0								0	1	0	0				
NISL3		24						0					0								0	0	1	0				
	PASI	25		1	0			1													0	0	0	0				
NISL2		26						1													0	0	1	0	1			
	PASO	27			0			0					0	0	0	1					0	0	1	0				
NISL2		28		1				0					1								0	1	0	0				
NISL1	SET PA	29						1					0								0	1	0	0				
	SET PA	30	1			0	1										1	0	0	0	1				0	0		
	DMP PA	31	1			0	0										1	0	0	0	1				0	0	0	
NISL0		32		1				0					1								0	1	0	0				
	WRITE	33						0					1								1	0	1					
	WRITE	34		0				0					1								1	1	0	0				
	WRITE	35				1	0	1																	0	1		
	WRITE	36		0		1	0	1																	1	0	0	
	SMC	37						1					0								0	1	0	0				
	SMC	38	1			0							1	0											0	0	0	
	SMC	39	1			0							0	1											0	1	0	
	SET IV	40	1			0	1										1	0	0	0	0				0	0		

X DENOTES A DIRECT CONNECTION BETWEEN STOPQ AND ABG

FIG 25-1-S

CONDENSED TABLE OF THE INSTRUCTION CONTROLLER
AND BUS CONTROLLER ROMS IN THE IOC

MICRO-INSTRUCTIONS			QUALIFIERS																										
STATE-COUNT	OTHER	OUTPUT LINE #	MEMORY & I/O CONTROL				QUALIFIER MULTIPLEXER						MACHINE INSTRUCTION				MEMORY ADDRESS					INSTRUCTION STATE-COUNT				BUS STATE-COUNT			
			STMQ	MECQ	STOPQ	DMAGQ	R/WQ OR	DR/WQ	INTQ OR	PCMQ	SYNCQ OR	CTMQ	IB6Q	IB5Q	IB4Q	IB3Q	IREGQ	PREGQ	ADB2Q	ADB1Q	ADB0Q	ISL3Q	ISL2Q	ISL1Q	ISL0Q	BSL2Q	BSL1Q	BSL0Q	
	SET DMAPA	41	1			0	1											1	0	0	1	1					0	0	
NISL1		42							0				1										1	0	1	0			
	DMP DMAPA	43	1			0	0											1	0	0	1	1					0	0	0
NISL0		44							0				1										1	0	1	0			
	DMP IV	45							1				0										0	1	0	0			
	DMP IV	46	1			0	0											1	0	0	0	0					0	0	0
	STM	47							0				1										0	0	1	1			
	STM	48							0				1										1	0	1	0			
	STM	49				1	1		1																		0	0	1
	STM	50				1	0		1																		0	1	0
	IC2	51	1			0												0	1		1	0							
	IC2	52	1			0												0	1		1	1							
	IC2	53				1					1																0	1	
	IC2	54				1					1																1	0	
	SET DMAMA	55	1			0	1											1	0	1	0	0					0	0	
NBSL2		56		0		1	0		1																		1	0	0
NBSL0	UPD DMAR	57				1																					0	0	0
NISL2		58		0					0				1										1	1	0	0			
	DMP DMAMA	59	1			0	0											1	0	1	0	0					0	0	0
	DMP DMAMA	60				1																					0	0	0
	DMP DMAC	61	1			0	0											1	0	1	0	1					0	0	0
NBSL0		62	1			0													1								0	0	0
	IC1	63	1			0												0	1		0	1							
	IC1	64	1			0												0	1		1	1							
	SET DMAC	65	1			0	1											1	0	1	0	1					0	0	
	BE	66				1			1																		1	0	0
	BE	67				1	1		1																		1	0	1
	BE	68	1			0												1	1			0					0		
	BE	69	1			0												0	1								0		
	BE	70		1		1	1		1																		0	1	0
	BE	71				1			1																		0	1	1
NBSL1		72				1			0																		0	0	1
NBSL1		73				0												1	1			0					0	0	1
NBSL1		74				0												1	1	0							0	1	0
	DMP DMAPA	75	1			0	0											1	0	0	1	1					0	0	0
NBSL2		76				0												1	1			0					0	1	1
	DIN	77	1			0												1	1			0					0		
	DIN	78	1			0	0											0	1										
	DIN	79				1	0		1																		0	0	1
	DIN	80				1	0		1																		0	1	

FIG 25-2-S

CONDENSED TABLE OF THE INSTRUCTION CONTROLLER
AND BUS CONTROLLER ROMS IN THE IOC

MICRO-INSTRUCTIONS			QUALIFIERS																								
STATE-COUNT	OTHER	OUTPUT LINE #	MEMORY & I/O CONTROL				QUALIFIER MULTIPLEXER				MACHINE INSTRUCTION					MEMORY ADDRESS					INSTRUCTION STATE-COUNT				BUS STATE-COUNT		
			STMQ	MECQ	STOPQ	DMAGQ	R/WQ OR DR/WQ	INTQ OR PCMQ	IB11Q	IB6Q	IB5Q	IB4Q	IB3Q	IREGQ	PREGQ	ADB2Q	ADB1Q	ADB0Q	ISL3Q	ISL2Q	ISL1Q	ISL0Q	BSL2Q	BSL1Q	BSL0Q		
	DIN	81				1	0	1															1	0			
	EBYT	82						0	0	1	1	0								1							
NISL2	UPD D	83						0		1			1							0	0	1	1				
NBSL1		84				0								1	1			0					0	1	1		
NBSL2		85				0								1	1			0		0			1	1	0		
	SET D	86	1			0	1							1	0	1	1	1					0	0			
	DMP DBY	87			0			0	0	1		0	1							0	1	0	1				
	DMP DBY	88			0			0	0	1		1	1							0	0	1	0				
NBSL2		89	1			0								1	0								0	0	1		
	DMP DWD	90			0			0	1	1		0	1							0	1	0	1				
	DMP DWD	91			0			0	1	1		1	1							0	0	1	0				
	DMP D	92	1			0	0							1	0	1	1	1					0	0	0		
	DMP CWD	93			0			0	1	1		0	0							0	1	0	1				
	DMP CWD	94			0			0	1	1		1	0							0	0	1	0				
	DMP C	95	1			0	0							1	0	1	1	0					0	0	0		
NBSL0		96	1			0								0	1								0	1	0		
	DMP CBY	97			0			0	0	1		0	0							0	1	0	1				
	DMP CBY	98			0			0	0	1		1	0							0	0	1	0				
	SET C	99	1			0	1							1	0	1	1	0					0	0			
NBSL2		100	1			0								1	1			1					0	0	1		
NBSL2		101	1			0								0	1								0	1	1		
NBSL0		102		1		1	1	1															0	1	0		
NISL2	UPD C	103						0	1	0										0	0	1	1				
NBSL0		104				1	0	1															0	1	0		
NBSL1		105				1		0															0	1	0		
NBSL0		106				1		0															1	0	0		
	IOSB	107	1			1	1	1															0	1	0		
	IOSB	108	1			0	0							0	1								0	1			
	IOSB	109	1			0	1							0	1								0	0	1		
	IOSB	110	1			0	1							0	1								0	1	0		
	IOSB	111				1																	0	1	1		
	IOSB	112				1																	1	0	0		
NBSL0		113				1	1	1															1	0	0		
	EPR	114				0								1	1			0					1	1	0		
NBSL2		115				1																	0	1	1		
NISL1	SET W	116			0			1	0											0	0	1	0	0			
NISL2	SET W	117		0				0	1											0	1	0	0				
	SET W	118	1			0	1							1	0	0	1	0					0	0			
	DMP ISC	119						1	0											0	1	0	0				
NBSL0		120	1			0								1									0	0	0		

CONDENSED TABLE OF THE INSTRUCTION CONTROLLER
AND BUS CONTROLLER ROMS IN THE IOC

MICRO-INSTRUCTIONS			QUALIFIERS																								
STATE-COUNT	OTHER	OUTPUT LINE #	MEMORY & I/O CONTROL				QUALIFIER MULTIPLEXER				MACHINE INSTRUCTION					MEMORY ADDRESS					INSTRUCTION STATE-COUNT				BUS STATE-COUNT		
			STMQ	MECQ	STOPQ	DMAGQ	R/WQ OR DR/WQ	INTQ OR PCMQ	IB11Q	IB6Q	IB5Q	IB4Q	IB3Q	IREGQ	PREGQ	ADB2Q	ADB1Q	ADB0Q	ISL3Q	ISL2Q	ISL1Q	ISL0Q	BSL2Q	BSL1Q	BSL0Q		
	DMP W	121						0			1								1	0	1	1					
	DMP W	122	1			0	0							1	0	0	1	0					0	0	0		
	SET I	123		0	0														0	0	0	0					
	SET I	124	1			0								1	1			1					0	0			
NISL1	DMP I	125			0			0			1		0						0	0	1	0					
	DMP I	126			0			0			1		0						0	1	0	1					
	DMP IDA	127			0			1			0								0	0	1	0	0				
	DMP IDA	128		0				0			1								0	1	0	0					
	DMP IDA	129		0	0			0						0					0	0	0	0					
	DMP IDA	130	1			0	1							1	0								0	0			
	DMP IDA	131	1			0								1	1			1					0	0			
	DMP IDA	132		0		1	1	1															0	1	0		
NBSL0		133		1		1	0	1															1	0	0		
NISL0	SET O	134						1			0								0	1	0	0					
NISL0	SET O	135			0			0			1								0	0	1	0					
NISL3	SET O	136			0			0			1								0	1	0	1					
NISL2	SET O	137						0			1								1	0	1	1					
	SET O	138	1			0	0							1	0								0	0	0		
	SET O	139				1																	0	0	0		
NBSL1	SET O	140		0		1	1	1															0	1	0		
	SIA	141			0			1			0								0	0	1	0					
NBSL0		142				1		0															0	1	0		
	SET IDA	143						1			0	0	0	0					0	1							
	SET IDA	144			0			0			1								0	0	1						
NISL1	SET IDA	145			0			0			1								0	1	0	1					
NISL3	SET IDA	146						0			1								1	0	1						
NISL3	SET IDA	147		0				0			1								1	1	0	0					
	SET IDA	148	1			0	0							1	0								0	0			
	SET IDA	149				1																	0	0	0		
NBSL1	SET IDA	150				1		1															0	0	1		
NBSL1	SET IDA	151		1		1	1	1															0	1	0		
NBSL1	SET IDA	152				1	0	1															0	1	0		
	SET IDA	153				1	1	1															0	1	1		
	SET IDA	154				1	1	1															1	0			
NISL0		155			0			0			1		1						0	0	1	0					
NBSL1		156				0								1	1			0	0				1	1	0		
NBSL1		157	1			0								0	1								0	1	0		
NBSL1		158	1			0								0	1								0	0	1		
	UPD CDD	159																	0	0	0	1					

FIG 25-4-S

CONDENSED TABLE OF THE INSTRUCTION CONTROLLER
AND BUS CONTROLLER ROMS IN THE IOC

MICRO-INSTRUCTIONS			QUALIFIERS																									
STATE-COUNT	OTHER	OUTPUT LINE #	MEMORY & I/O				QUALIFIER						MACHINE				MEMORY					INSTRUCTION				BUS		
			CONTROL				MULTIPLEXER						INSTRUCTION				ADDRESS					STATE-COUNT				STATE-COUNT		
			STMQ	MECQ	STOPQ	DMAGQ	R/WQ OR	DR/WQ	INTQ OR	PCMQ	SYNCQ OR	CTMQ	IB6Q	IB5Q	IB4Q	IB3Q	IREGQ	PREGQ	ADB2Q	ADB1Q	ADB0Q	ISL3Q	ISL2Q	ISL1Q	ISL0Q	BSL2Q	BSL1Q	BSL0Q
	EPCMR	1			0				0			0	1	0	1							0	0	1	0			
NISL0		2		1																		0	0	0	0			
	SYNC	3		0																		0	0	0	0			
	SYNC	4			0				0				0	0	0	0						0	0	1	0			
	SYNC	5											0									1	0	0	0			
	SYNC	6		1					0				1									1	1	0	0			
	ABG	7																					0	0	0			
NISL1	ABG	8																				0	0	0	1			
	ABG	9											0									0	0	1	0			
	ABG	10											0									0	1	0	0			
	ABG	11		1					0				1										1	0	0			
	ABG	12			X																							
NBSL2	UBG	13				1	1	1																		1	0	0
NBSL2	UBG	14		1		1	0	1																		1	0	0
NBSL2	UBG	15				1		0																		1	0	0
	UINTR	16			0			0					0	0	1							0	0	1	0			
	EINTR	17			0			0					0	0	1	0						0	0	1	0			
NISL2		18						1														0	0	1	0	1		
	UDMAR	19			0			0					0	1								0	0	1	0			
NISL3		20						1														0	1	1	1			
	EDMAR	21			0			0					0	1	0							0	0	1	0			
NISL3		22								0			0									1	0	0	0			
NISL2	UIG	23						1					0									0	1	0	0			
NISL3		24						0					0									0	0	1	0			
	PASI	25		1	0			1														0	0	0	0			
NISL2		26						1														0	0	1	0	1		
	PASO	27			0			0					0	0	0	1						0	0	1	0			
NISL2		28		1				0					1									0	1	0	0			
NISL1	SET PA	29						1					0									0	1	0	0			
	SET PA	30	1			0	1										1	0	0	0	1					0	0	
	DMP PA	31	1			0	0										1	0	0	0	1					0	0	0
NISL0		32		1				0					1									0	1	0	0			
	WRITE	33						0					1									1	0	1				
	WRITE	34		0				0					1									1	1	0	0			
	WRITE	35				1	0	1																		0	1	
	WRITE	36		0		1	0	1																		1	0	0
	SMC	37						1					0									0	1	0	0			
	SMC	38	1			0											1	0								0	0	0
	SMC	39	1			0											0	1								0	1	0
	SET IV	40	1			0	1										1	0	0	0	0					0	0	

X DENOTES A DIRECT CONNECTION BETWEEN STOPQ AND ABG

FIG 25-1-F

CONDENSED TABLE OF THE INSTRUCTION CONTROLLER
AND BUS CONTROLLER ROMS IN THE IOC

MICRO-INSTRUCTIONS			QUALIFIERS																									
STATE-COUNT	OTHER	OUTPUT LINE #	MEMORY & I/O CONTROL				QUALIFIER MULTIPLEXER					MACHINE INSTRUCTION				MEMORY ADDRESS					INSTRUCTION STATE-COUNT				BUS STATE-COUNT			
			STMQ	MECQ	STOPQ	DMAGQ	R/WQ OR	DR/WQ	INTQ OR	PCMQ	SYNCO OR	CTMQ	IB6Q	IB5Q	IB4Q	IB3Q	IREGQ	PREGQ	ADB2Q	ADB1Q	ADB0Q	ISL3Q	ISL2Q	ISL1Q	ISL0Q	BSL2Q	BSL1Q	BSL0Q
	SET DMAPA	41	1			0	1										1	0	0	1	1					0	0	
NISL1		42							0				1									1	0	1	0			
	DMP DMAPA	43	1			0	0										1	0	0	1	1					0	0	0
NISL0		44							0				1									1	0	1	0			
	DMP IV	45							1				0									0	1	0	0			
	DMP IV	46	1			0	0										1	0	0	0	0					0	0	0
	STM	47							0				1									0	0	1	1			
	STM	48							0				1									1	0	1	0			
	STM	49				1	1	1																		0	0	1
	STM	50				1	0	1																		0	1	0
	IC2	51	1			0											0	1		1	0							
	IC2	52	1			0											0	1		1	1							
	IC2	53				1					1															0	1	
	IC2	54				1					1															1	0	
	SET DMAMA	55	1			0	1										1	0	1	0	0					0	0	
NBSL2		56		0		1	0	1																		1	0	0
NBSL0	UPD DMAR	57				1																				0	0	0
NISL2		58		0					0				1									1	1	0	0			
	DMP DMAMA	59	1			0	0										1	0	1	0	0					0	0	0
	DMP DMAMA	60				1																				0	0	0
	DMP DMAC	61	1			0	0										1	0	1	0	1					0	0	0
NBSL0		62	1			0												1								0	0	0
	IC1	63	1			0											0	1		0	1							
	IC1	64	1			0											0	1		1	1							
	SET DMAC	65	1			0	1										1	0	1	0	1					0	0	
	NOT USED	66																										
	BE	67				1	1	1																		1	0	1
	BE	68	1			0											1	1			0					0		
	BE	69	1			0											0	1								0		
	BE	70		1		1	1	1																		0	1	0
	BE	71				1			1																	0	1	1
	BE	72				1			1																	1	0	0
NBSL1		73				0											1	1			0					0	0	1
NBSL1		74				0											1	1			0					0	1	0
NBSL0		75				0											1	1			0					0	1	0
NBSL2		76				0											1	1			0					0	1	1
	DIN	77	1			0											1	1			0					0		
	DIN	78	1			0	0										0	1										
	DIN	79				1	0	1																		0	0	1
	DIN	80				1	0	1																		0	1	

FIG 25-2-F



CONDENSED TABLE OF THE INSTRUCTION CONTROLLER
AND BUS CONTROLLER ROMS IN THE IOC

MICRO-INSTRUCTIONS			QUALIFIERS																									
STATE- COUNT	OTHER	OUTPUT LINE #	MEMORY & I/O				QUALIFIER						MACHINE				MEMORY					INSTRUCTION				BUS		
			CONTROL				MULTIPLEXER						INSTRUCTION				ADDRESS					STATE-COUNT				STATE-COUNT		
			STMQ	MECQ	STOPQ	DMAGQ	R/WQ OR	DR/WQ	INTQ OR	PCMQ	SYNCQ OR	CTMQ	IB6Q	IB5Q	IB4Q	IB3Q	IREGQ	PREGQ	ADB2Q	ADB1Q	ADB0Q	ISL3Q	ISL2Q	ISL1Q	ISL0Q	BSL2Q	BSL1Q	BSL0Q
	DIN	81				1	0		1																1	0		
	EBYT	82							0			1	1	0							1							
NISL2	UPD C	83							0			1			0						0	0	1	1				
NBSL1		84				0										1	1			0				0	1	1		
NISL2	UPD D	85							0			1			1						0	0	1	1				
NBSL2		86				0										1	1			0	0			1	1	0		
NBSL1		87				0										1	1			0	0			1	1	0		
NBSL1		88	1			0										0	1							0	1	0		
NBSL1		89	1			0										0	1							0	0	1		
NBSL0		90	1			0										0	1							0	1	0		
NBSL2		91	1			0										0	1							0	1	1		
NBSL2		92	1			0										1	0							0	0	1		
	SET C	93	1			0	1									1	0	1	1	0				0	0			
NBSL2		94	1			0										1	1			1				0	0	1		
NBSL1		95				1			0															0	0	1		
	DMP C	96	1			0	0									1	0	1	1	0				0	0	0		
	DMP C	97			0				0			1		0	0						0	1	0	1				
NISL1	DMP C	98			0				0			1		1	0						0	0	1	0				
	DMP D	99	1			0	0									1	0	1	1	1				0	0	0		
	DMP D	100			0				0			1		0	1						0	1	0	1				
NISL1	DMP D	101			0				0			1		1	1						0	0	1	0				
NBSL0		102		1		1	1		1															0	1	0		
	SET D	103	1			0	1									1	0	1	1	1				0	0			
NBSL0		104				1	0		1															0	1	0		
NBSL1		105				1			0															0	1	0		
NBSL0		106				1			0															1	0	0		
	IOSB	107	1			1	1		1															0	1	0		
	IOSB	108	1			0	0									0	1							0	1			
	IOSB	109	1			0	1									0	1							0	0	1		
	IOSB	110	1			0	1									0	1							0	1	0		
	IOSB	111				1																		0	1	1		
	IOSB	112				1																		1	0	0		
NBSL0		113				1	1		1															1	0	0		
	EPR	114				0										1	1			0				1	1	0		
NBSL2		115				1																		0	1	1		
NISL1	SET W	116			0				1			0									0	0	1	0	0			
NISL2	SET W	117			0				0			1									0	1	0	0				
	SET W	118	1			0	1									1	0	0	1	0				0	0			
	DMP ISC	119							1			0									0	1	0	0				
NBSL0		120	1			0										1								0	0	0		
	DMP W	121							0			1									1	0	1	1				
	DMP W	122	1			0	0									1	0	0	1	0				0	0	0		
	SET I	123			0	0															0	0	0	0				
	SET I	124	1			0										1	1			1				0	0			
NISL1	DMP I	125			0				0			1		0							0	0	1	0				
	DMP I	126			0				0			1		1							0	1	0	1				
	DMP IDA	127			0				1			0									0	0	1	0	0			
	DMP IDA	128			0				0			1									0	1	0	0				
	DMP IDA	129			0	0			0							0					0	0	0	0				
	DMP IDA	130	1			0	1									1	0							0	0			

FIG 25-3-F

CONDENSED TABLE OF THE INSTRUCTION CONTROLLER AND BUS CONTROLLER ROMS IN THE IOC

MICRO-INSTRUCTIONS			QUALIFIERS																									
STATE-COUNT	OTHER	OUTPUT LINE #	MEMORY & I/O CONTROL				QUALIFIER MULTIPLEXER						MACHINE INSTRUCTION				MEMORY ADDRESS					INSTRUCTION STATE-COUNT				BUS STATE-COUNT		
			STMQ	MECQ	STOPQ	DMAGQ	R/WQ OR DR/WQ	INTQ OR PCMQ	SYNCO OR CTMQ	IB6Q	IB5Q	IB4Q	IB3Q	IREGQ	PREGQ	ADB2Q	ADB1Q	ADB0Q	ISL3Q	ISL2Q	ISL1Q	ISL0Q	BSL2Q	BSL1Q	BSL0Q			
	DMP IDA	131	1			0								1	1			1					0	0				
	DMP IDA	132		0		1	1	1															0	1	0			
NBSL0		133		1		1	0	1															1	0	0			
NISL0	SET O	134						1		0									0	1	0	0						
NISL0	SET O	135			0			0		1									0	0	1	0						
NISL3	SET O	136			0			0		1									0	1	0	1						
NISL2	SET O	137						0		1									1	0	1	1						
	SET O	138	1			0	0							1	0								0	0	0			
	SET O	139				1																	0	0	0			
NBSL1	SET O	140		0		1	1	1															0	1	0			
	SIA	141			0			1		0									0	0	1	0						
NBSL0		142				1		0															0	1	0			
	SET IDA	143						1		0	0	0	0						0	1								
	SET IDA	144			0			0		1									0	0	1							
NISL1	SET IDA	145			0			0		1									0	1	0	1						
NISL3	SET IDA	146						0		1									1	0	1							
NISL3	SET IDA	147		0				0		1									1	1	0	0						
	SET IDA	148	1			0	0							1	0								0	0				
	SET IDA	149				1																	0	0	0			
NBSL1	SET IDA	150				1		1															0	0	1			
NBSL1	SET IDA	151		1		1	1	1															0	1	0			
NBSL1	SET IDA	152				1	0	1															0	1	0			
	SET IDA	153				1	1	1															0	1	1			
	SET IDA	154				1	1	1															1	0				
	NOT USED	155																										
	NOT USED	156																										
	NOT USED	157																										
	NOT USED	158																										
	NOT USED	159																										
	NOT USED	160																										

FIG 25-4-F

NOTES ON THE CONDENSED TABLE

- THE TABLE DEPICTS WHAT "AND" CONDITIONS MUST BE MET FOR A GIVEN MICRO-INSTRUCTION TO BE DECODED. A ONE IN A QUALIFIER COLUMN DENOTES THAT THE STATED QUALIFIER MUST BE TRUE IN ORDER FOR THE MICRO-INSTRUCTION TO BE GIVEN. A ZERO INDICATES THE QUALIFIER MUST BE FALSE. A BLANK DENOTES THAT THE QUALIFIER DOES NOT AFFECT THE DECODING OF THE MICRO-INSTRUCTION.
- ONE'S IN THE QUALIFIER COLUMNS ARE ELECTRICALLY IMPLEMENTED IN THE ROM BY CONNECTING THE GATES OF DECODING TRANSISTORS ON THE VARIOUS HORIZONTAL OUTPUT LINES TO A VERTICAL DRIVE LINE THAT REPRESENTS THE NOT OF THAT QUALIFIER. ZEROS ARE IMPLEMENTED BY CONNECTING THE GATES OF THE DECODING TRANSISTORS TO THE TRUE SENSE OF THE QUALIFIER. A BLANK DENOTES THE TOTAL ABSENSE OF A DECODING TRANSISTOR.
- A HORIZONTAL OUTPUT LINE CAN REPRESENT BOTH A STATE-COUNT CHANGE MICRO-INSTRUCTION AND AN ORDINARY MICRO-INSTRUCTION. IN SUCH A CASE, ONE (PHYSICAL) END OF THE LINE IS CONNECTED TO THE APPROPRIATE NEXT STATE-COUNT ENCODER, WHILE THE OTHER END GOES TO THE ORDINARY MICRO-INSTRUCTION FAN-IN MECHANISM.

FIG 25-5

ROM BIT PATTERN

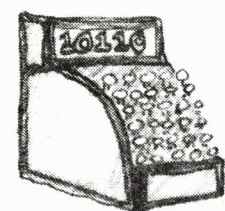
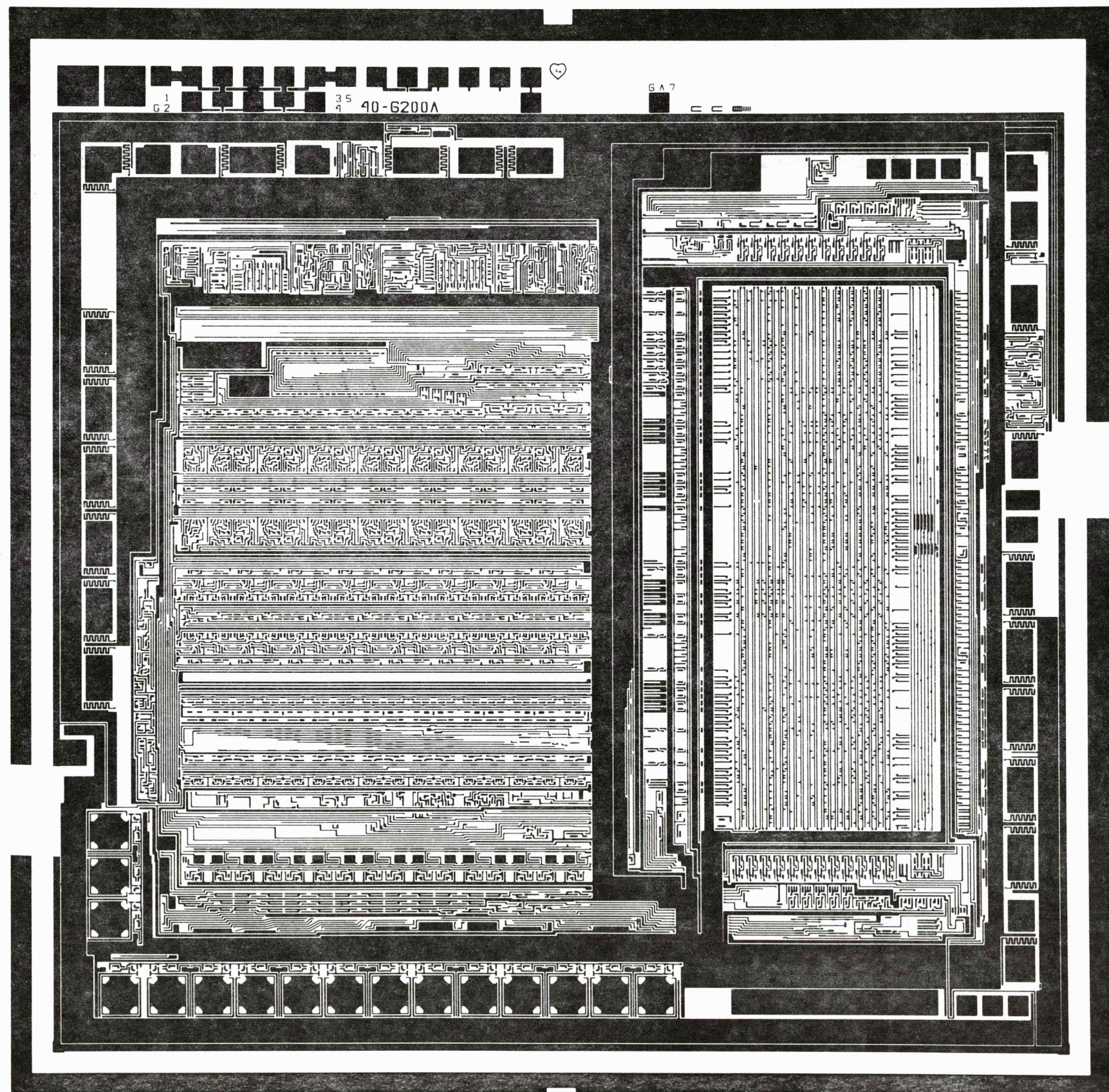


ROM BIT PATTERN



ROM BIT PATTERN
NOTES

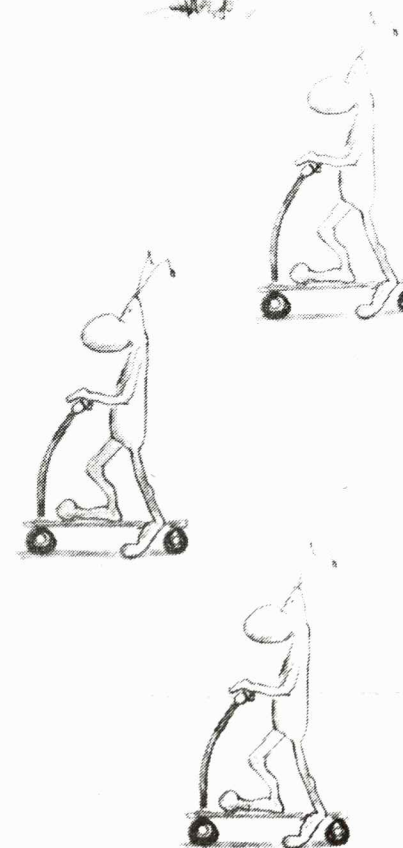
40-6200A-6A



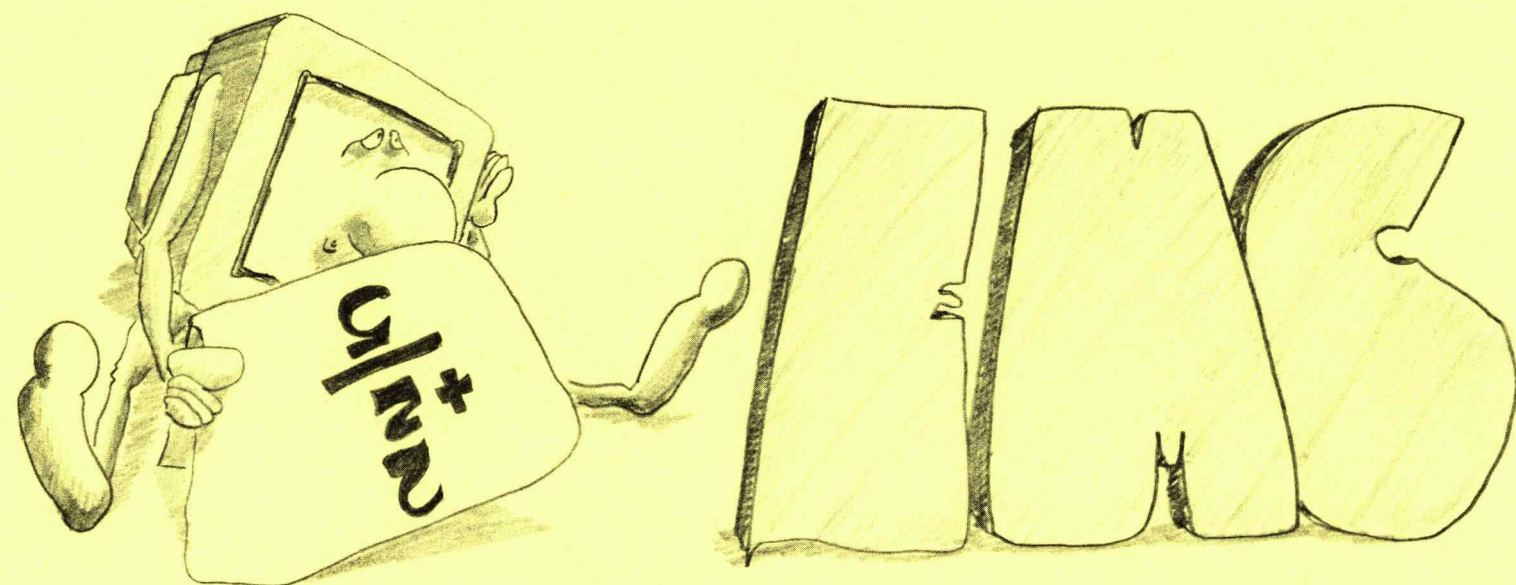
"REGISTER"



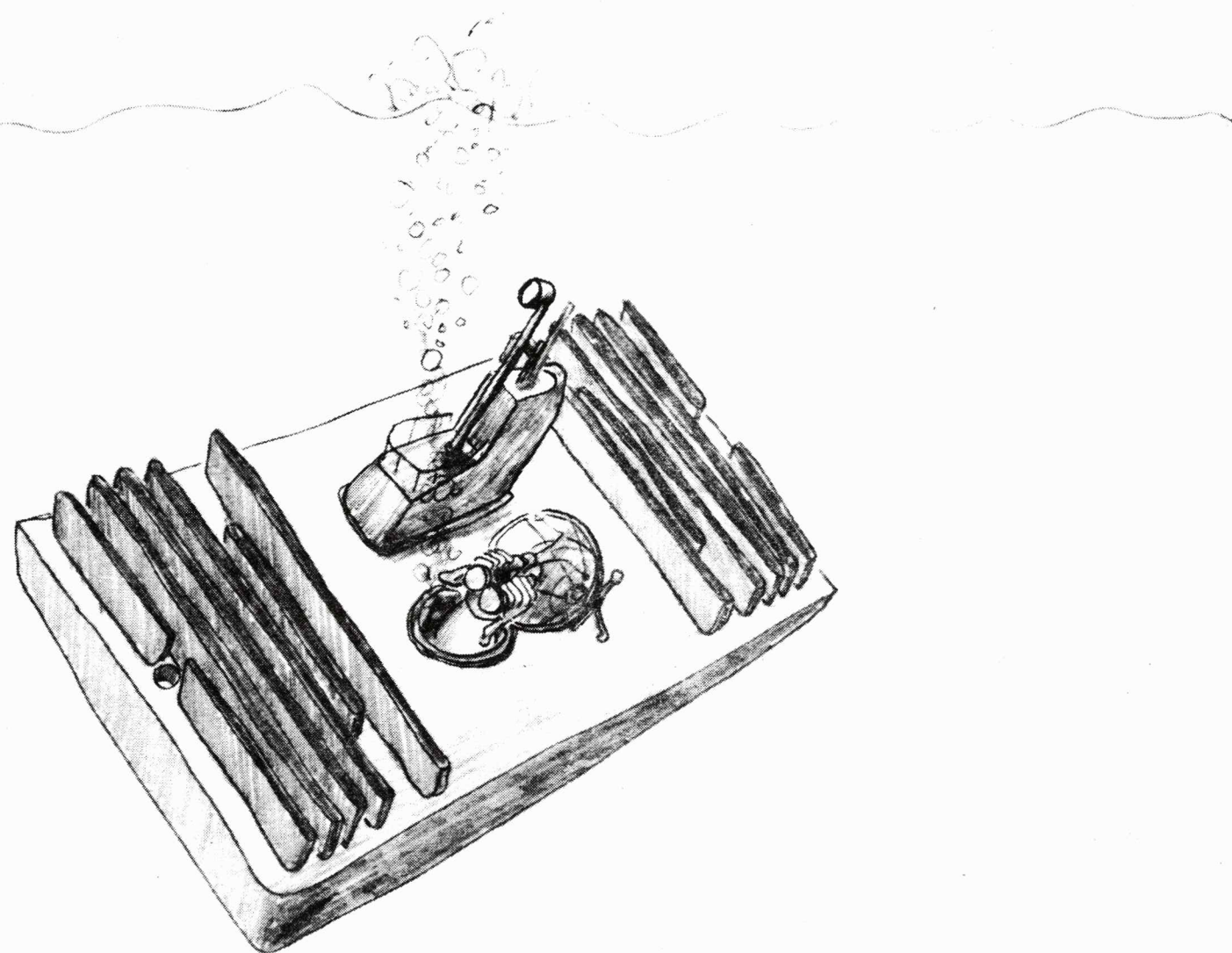
111 1115E



3-3-76



2/5/78



AT FIRST, THERE WERE A FEW BUGS...

SECTION 1

Figure 1 depicts the internal block diagram of the EMC. The micro-instructions SET IDA and DMP IDA are the communication link between the external IDA Bus and the internal IDM Bus. The D register is used as a source for data for a SET IDA. Instructions are fetched by the BPC and placed on the IDA Bus. The EMC decodes them and then reacts accordingly.

If a fetched instruction is not an EMC instruction, or if an interrupt occurs, the EMC ignores the fetched instruction. (In the latter case, it will be fetched again at the conclusion of the interrupt service routine.) Upon completion of an instruction by another chip and in the absence of an interrupt, the EMC examines the next fetched instruction. If the instruction is an EMC instruction, it is executed and data affected by it are transferred via the IDA Bus. At the appropriate point during the execution of the instruction, SYNC is given to indicate to other chips that the EMC has finished using the IDA Bus, and consequently, to treat the next item that appears on the IDA Bus as an instruction. Section 14 contains flow charts detailing the manner in which the EMC executes its machine-instructions.

The main function of the EMC is to provide BCD arithmetic. To this end the EMC is equipped with a BCD/Binary Adder, a One's/Nine's Complementer, and two sets of multi-part registers to act as inputs to the Adder.

The Adder is an extension of the one used in the BPC. The signal BCD determines whether it adds in binary or in BCD. One of the multi-part registers (Y1-Y3) is connected to the Adder through the Complementer. These Y registers are also associated with the Y0 register which is an exponent word. Y0 is not connected to the Adder. Collectively, these registers are referred to as simply the Y register. The Y register is the same as AR2, and represents a "permanent" input to the Adder during BCD arithmetic machine-instructions. The other multi-part register (X1-X3) serves as the other input to the Adder. Collectively, these are called the X register. The X register is

used to hold an internal copy of AR1 during BCD arithmetic machine-instructions, but is not itself AR1. AR1 is in R/W memory.

As in the BPC, the Adder is only a 16-bit wide mechanism. The Adder in the EMC deals with the individual parts of the multi-part registers, one at a time, in sequence. A 1-bit Decimal Carry register (DC) serves to link one partial addition to the next. A Word Pointer Shift Register selects which segments of Y1-Y3 and X1-X3 are bussed to the Adder and Complementer.

The Word Pointer Shift Register points to the register to be affected by the DMPX/SETX or DMPY/SETY micro-instructions as well as specifying the registers to be bussed to the Adder. It is also employed as a counter in some instructions.

Once data is on the IDM Bus it can then be loaded into one of several registers by issuing the appropriate micro-instruction. The data paths between the IDM Bus and the X and Y registers can be controlled in two ways. One way is by issuing an explicit micro-instruction, e.g., SET Y2 would set the Y2 register with the data on IDM. Another way of accomplishing the same thing would be to issue a SET Y with the Word Pointer equalling two.

The X registers are used for all shifting operations: the direction of the shift is dependent upon the asynchronous control lines.

The Shift Extend register is a four-bit addressable register used to hold a digit to be shifted into the X register, or to hold one that has been shifted out of X.

The Arithmetic Extend register is a 4-bit addressable (read-only) register used to accumulate a decimal digit for the FMP and FDV instructions and serves as a number-of-shifts accumulator in the NRM instruction.

The N Counter is used: to indicate the number of words involved in the CLR and XFR instructions; to indicate the number of shifts in MRX, MRY, MLY and DRS; to contain the multiplier digit in FMP; and as a loop counter in MPY.

ADRI is the address of the AR1 operand; its two least significant

bits are determined by the word pointer, e.g., WPO implies 00, WPI implies 01, etc.

Miscellaneous hardware enhances the execution of the two's complement binary multiply instruction (MPY).

The M-Section of the EMC is responsible for EMC responses to all memory cycles which are directed to it, whether by outside agencies or by the EMC itself. (Memory cycles originated by the EMC to memory external to the EMC are all handled by the section of flow charting responsible.) The EMC M-Section is similar to that of the BPC, except that in the EMC the generation of the appropriate SET OR DUMP micro-instructions is under the control of


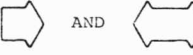

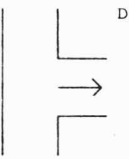
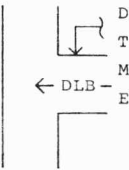

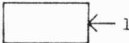



a separate ROM. That ROM is the Address Decode ROM. It supplies SET or DUMP micro-instructions based upon the address latched by the Register Detection/Address Latch circuit, and based on whether the memory cycle is a read or a write cycle.

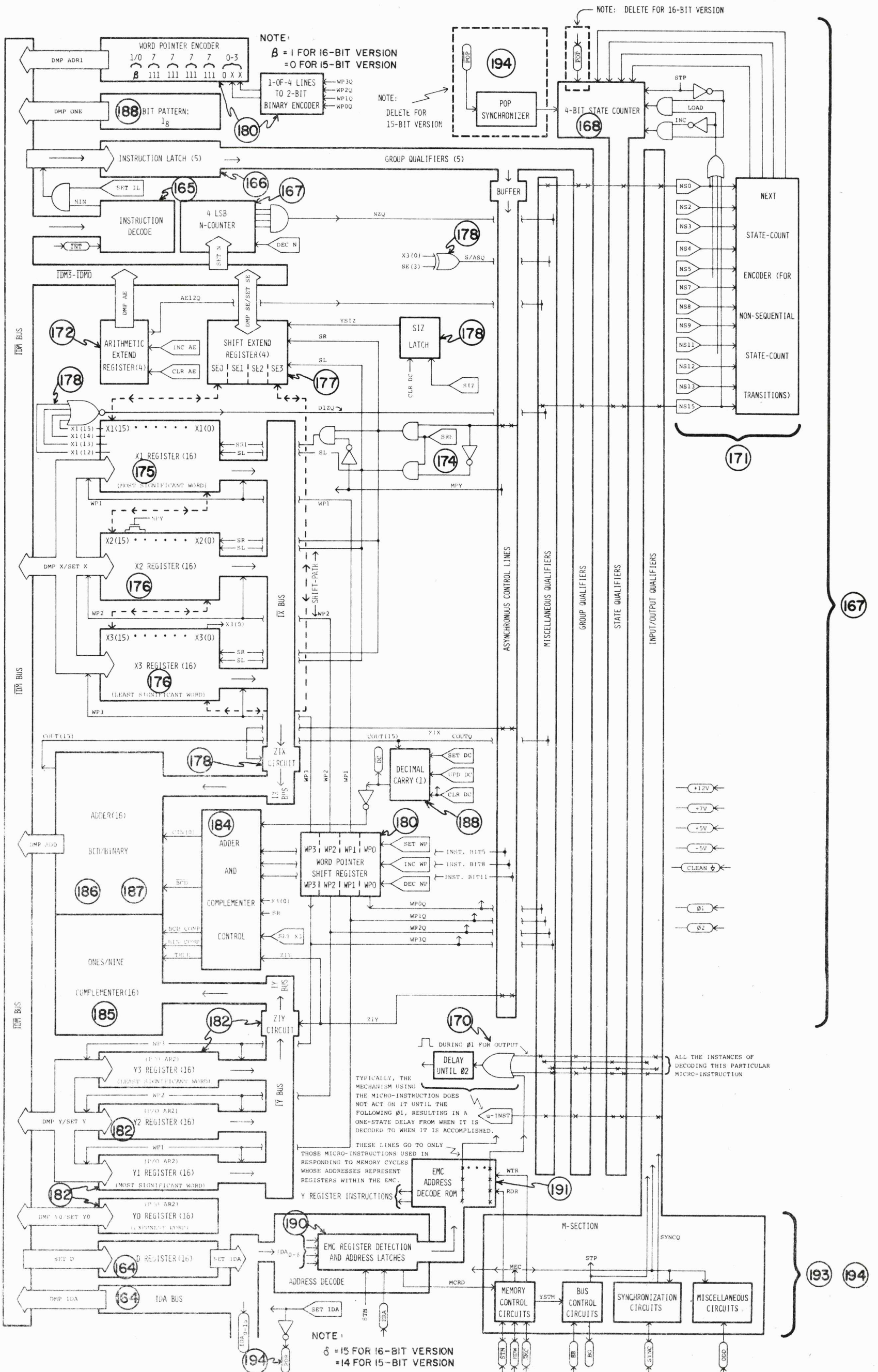
The differences between the 15 and 16-bit versions concern only some bugs, and the formation and recognition of addresses. The 16-bit version is generally considered compatible with 15-bit applications; the 15-bit version is obsolete.

Numbers in circles (XXX) →

refer to the page number at which the referenced item is discussed.

NOTES: FOR FIG. 1

1.  DENOTES A MICRO-INSTRUCTION DECODED IN THE ROM.
2.  AND  DENOTE ONE- AND TWO-WAY INTERCONNECTIONS TO A BUS; ALWAYS CONTROLLED BY A ROM MICRO-INSTRUCTION.
3.  DENOTES A DIRECT CONNECTION BETWEEN TWO ITEMS.
4.  DENOTES A CONNECTION BETWEEN TWO ITEMS THAT IS ACTIVE ONLY WHEN THE STATED SIGNAL IS GIVEN. SOME SUCH SIGNALS ARE ROM DECODED MICRO-INSTRUCTIONS, WHILE OTHERS ARE PRESENT THROUGHOUT AN ENTIRE EXECUTION CYCLE.
5.  DENOTES THAT THE STATED LINE REPRESENTS A DECODED CONDITION.
6.  REPRESENTS A NON-MICRO-INSTRUCTION CONTROL LINE OR SOME OTHER SIGNAL.
7.  REPRESENTS AN INPUT TERMINAL TO THE EMC.
8.  REPRESENTS AN OUTPUT TERMINAL FROM THE EMC.
9.  REPRESENTS A TERMINAL THAT IS BOTH AN INPUT AND AN OUTPUT.
10. NUMBERS IN PARENTHESES INDICATE THE NUMBER OF BITS A MECHANISM HANDLES.
11. THE LOGICAL SENSE (XXX VERSUS XXX) OF THE I/O TERMINALS IS CORRECTLY INDICATED. HOWEVER, THE DRAWING IS NOT A RELIABLE INDICATOR OF THE EXACT SENSE OF THE INTERNAL SIGNALS. TYPICALLY BOTH SENSES EXIST, AND FREQUENTLY THE PHYSICAL PROXIMITY OF SIGNALS TO THEIR DESTINATIONS WAS MORE IMPORTANT IN DECIDING WHICH SENSE TO USE, RATHER THAN AGREEMENT OF LOGICAL SENSE.
BECAUSE STRICT ACCURACY IN REPORTING SIGNAL SENSES ON SUCH A GENERAL LEVEL DRAWING WOULD SHARPLY INCREASE THE NUMBER OF INTERCONNECTIONS, WITH ONLY A SLIGHT INCREASE IN USEFULNESS, WE USUALLY SHOW ONLY THE NAME OF THE SIGNAL.



OVERVIEW OF THE IDA BUS - IDM BUS INTERCONNECTION

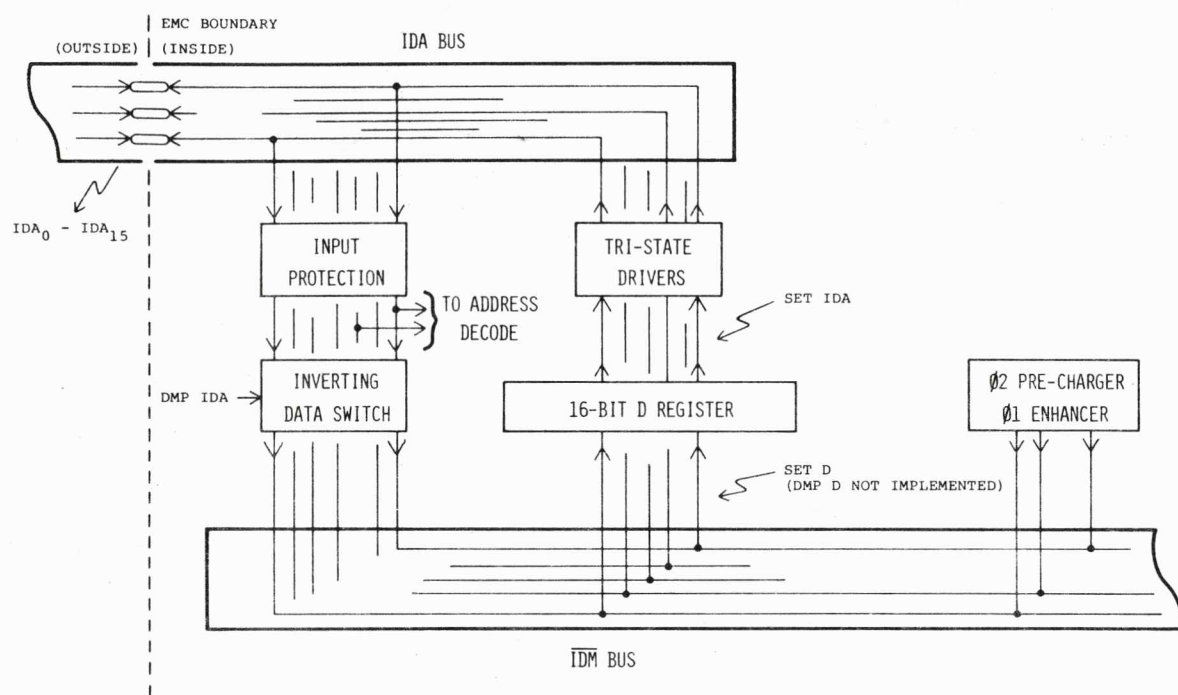


FIG 2-1

SECTION 2

Figure 2-1 is an overview of the connection between the external IDA Bus and the internal IDM Bus in the EMC. The purpose, as well as the actual form, of this circuitry is virtually identical to the corresponding circuitry in the BPC. Refer to the explanation of the BPC for more information.

DETAILS OF THE D REGISTER AND THE IDA BUS DRIVERS

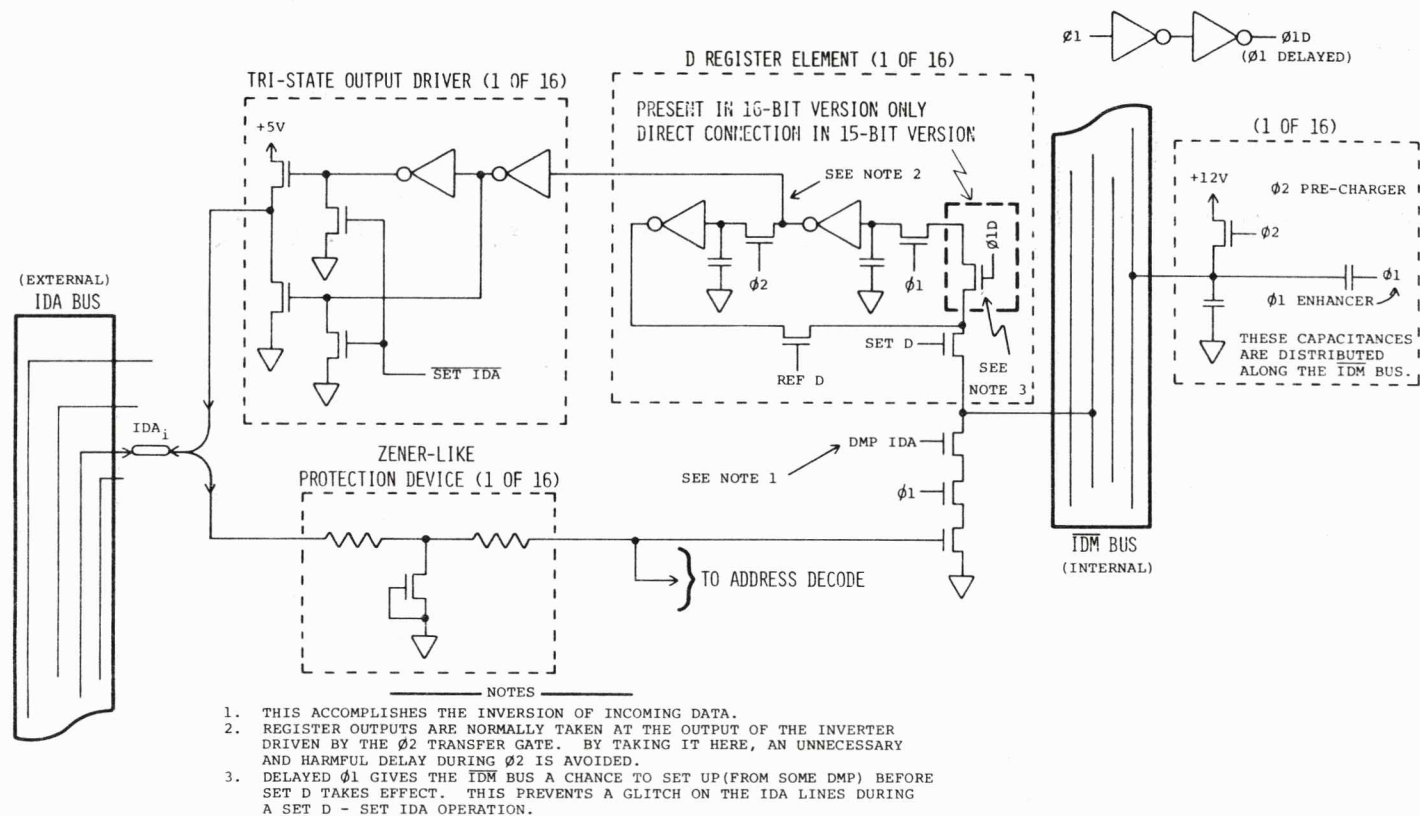


FIG 2-2

Figure 2-2 illustrates the details of the D register, IDA Bus Drivers actuated by SET IDA, and of the Phase One Enhancer. This circuitry is essentially identical to that of the BPC. Refer to the explanation of the BPC for more information.

OVERVIEW OF INSTRUCTION DECODE, INSTRUCTION LATCH AND N COUNTER.

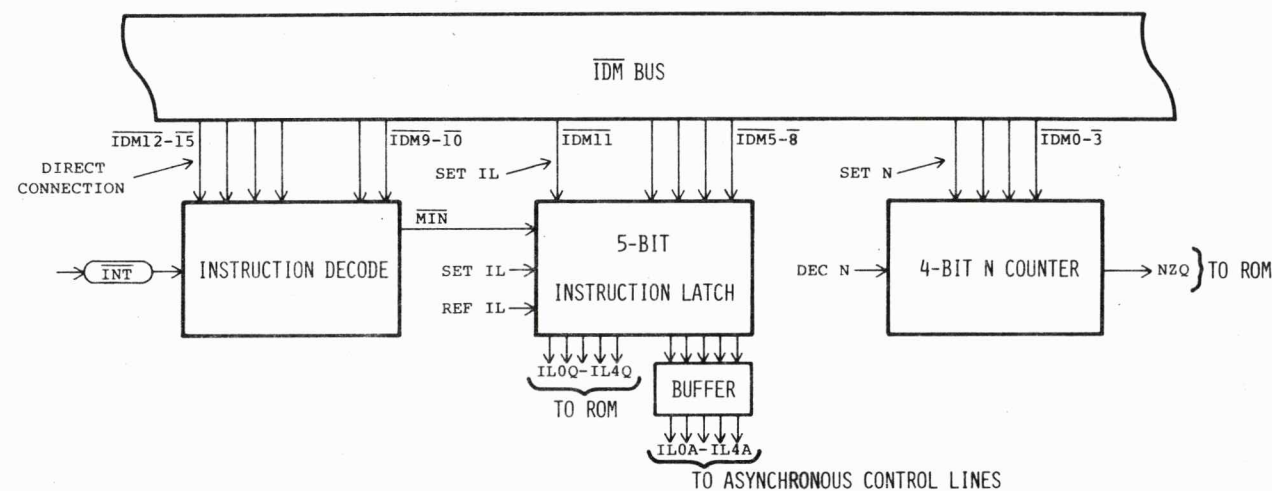


FIG 3-1

DETAILS OF INSTRUCTION DECODE

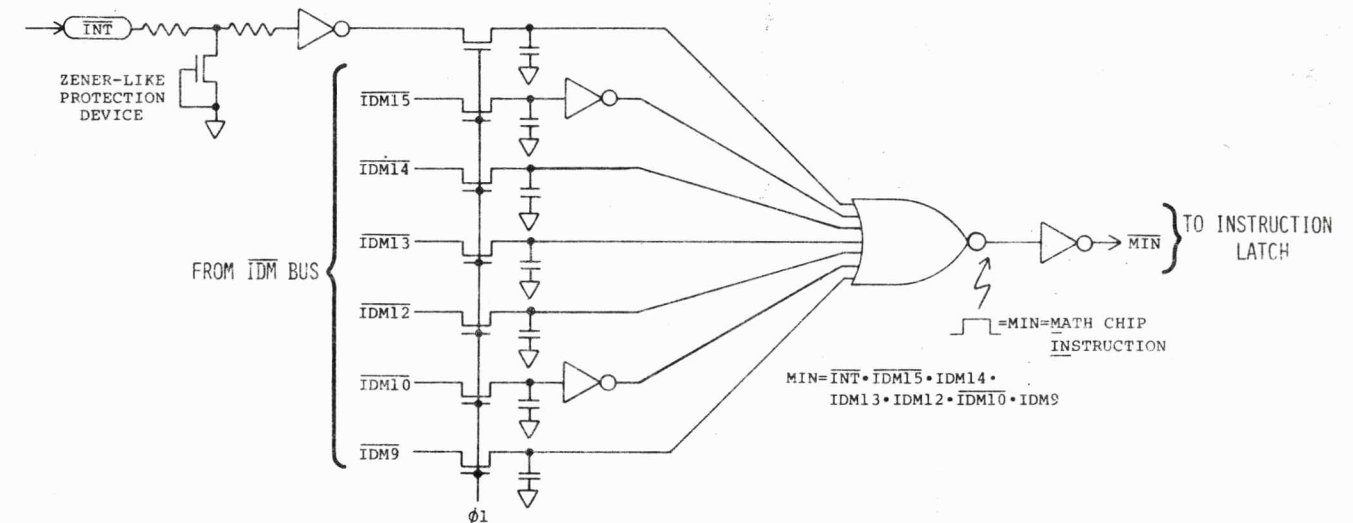


FIG 3-2-1

SECTION 3

Figure 3-1 is an overview of the relationship between Instruction Decode, the Instruction Latch and the N Counter. The purpose of this circuitry is to detect and capture machine-instruction bit patterns that are associated with the EMC.

During an instruction fetch the EMC ASM chart issues micro-instructions SET IL and SET N. During this time the machine-instruction bit pattern is on the IDB Bus. Instruction Decode monitors that bit pattern; if it is an EMC machine-instruction the signal MIN goes low.

Meanwhile, the SET IL is attempting to put the necessary bits of an EMC machine-instruction bit pattern into the Instruction Latch. It will be successful in this if MIN is low. Otherwise, a pattern of all ones will be loaded into the Instruction Latch. The pattern of all ones overrides the bit pattern associated with any non-EMC instruction, and is the explicit means used by the ASM chart to determine that the machine-instruction does not pertain to the EMC.

The Instruction Latch generates qualifiers for the ROM that control

which instruction group (major segment) of the ASM chart that is accessed. The Instruction Latch is also used in the generation of the asynchronous control lines, whose functions are similar to those of the BPC.

Some EMC machine-instructions have counts encoded in their least four significant bits. During an instruction fetch this count is put into the N Counter with SET N. (If the machine-instruction does not have an associated count the content of the N Counter is a don't-care.) The N Counter will be deliberately decrement-

ed by the flow charting each time an operation corresponding to the count is performed. The qualifier NZQ indicates a count of zero.

Figure 3-2-1 shows the details of Instruction Decode. Basically, it is simply a not'ed input NOR gate to detect a certain pattern on the IDB Bus. Observe that INT affects the operation of Instruction Decode. If INT is true, MIN will be false. In this way, the EMC will refuse to respond to any instruction fetched when an interrupt occurs.

INSTRUCTION
DECODE

INSTRUCTION DECODE,
INSTRUCTION LATCH &
N COUNTER OVERVIEW

D REGISTER
& 01 ENHANCER

IDA - IDB
OVERVIEW



DETAILS OF THE EMC INSTRUCTION BIT PATTERNS

THESE BITS DIFFERENTIATE THE VARIOUS INSTRUCTIONS FROM EACH OTHER AND ARE CAPTURED BY THE INSTRUCTION LATCH TO BECOME THE GROUP QUALIFIERS

THESE BITS ARE DON'T CARES EXCEPT AS SHOWN

INST. NAME	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FXA	0	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0
MWA	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0
CMX	0	1	1	1	0	0	1	0	0	1	1	0	0	0	0	0
CMY	0	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0
FMP	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0
FDV	0	1	1	1	1	0	1	0	0	0	1	0	0	0	0	1
MPY	0	1	1	1	1	0	1	1	1	0	0	0	1	1	1	1
CDC	0	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0
MRX	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
DRS	0	1	1	1	1	0	1	1	0	0	1	0	0	0	0	1
MLY	0	1	1	1	1	0	1	1	0	1	1	0	0	0	0	1
MRY	0	1	1	1	1	0	1	1	0	1	0	0	0	0	0	0
NRM	0	1	1	1	0	0	1	1	0	1	0	0	0	0	0	0
CLR	0	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0
XFR	0	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0

INSTANCES OF PRE-SETTING THE VALUE OF THE N COUNTER

* 4 BIT FIELD # OF WORDS
* BINARY = N-1

FIG 3-2-2

SECTION 3 (CONTINUED)

Figure 3-2-2 depicts the bit patterns of the machine-instructions associated with the EMC. It shows which bits are used to generate MIN, and the instances of presetting the value of the N Counter.

DETAILS OF THE INSTRUCTION LATCH

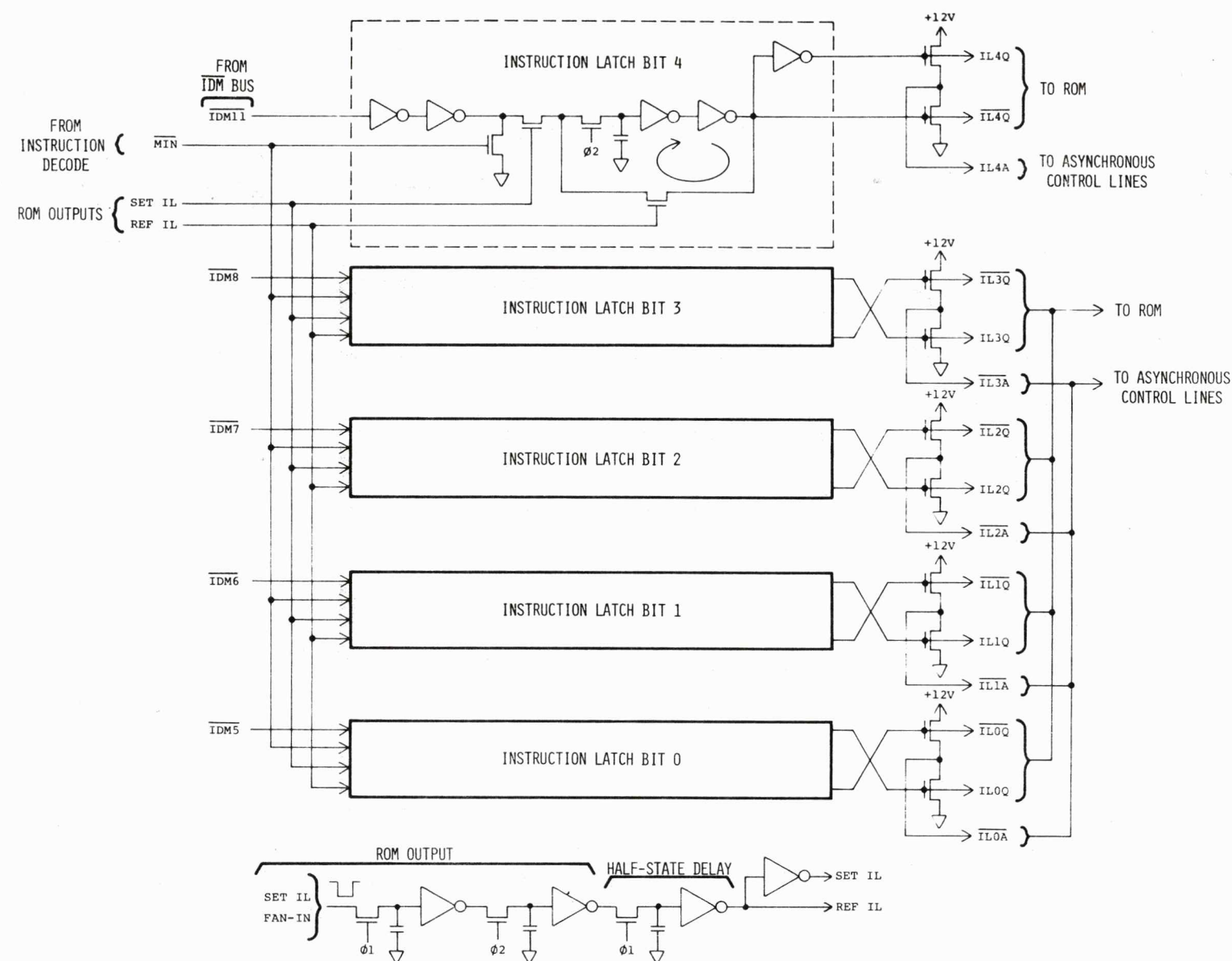


FIG 3-3

Figure 3-3 shows the details of the Instruction Latch and of the SET IL micro-instruction. The Instruction Latch is used to capture the bits that differentiate one EMC machine-instruction from another. The latches are enabled when MIN is false and SET IL is issued.

DETAILS OF THE N COUNTER

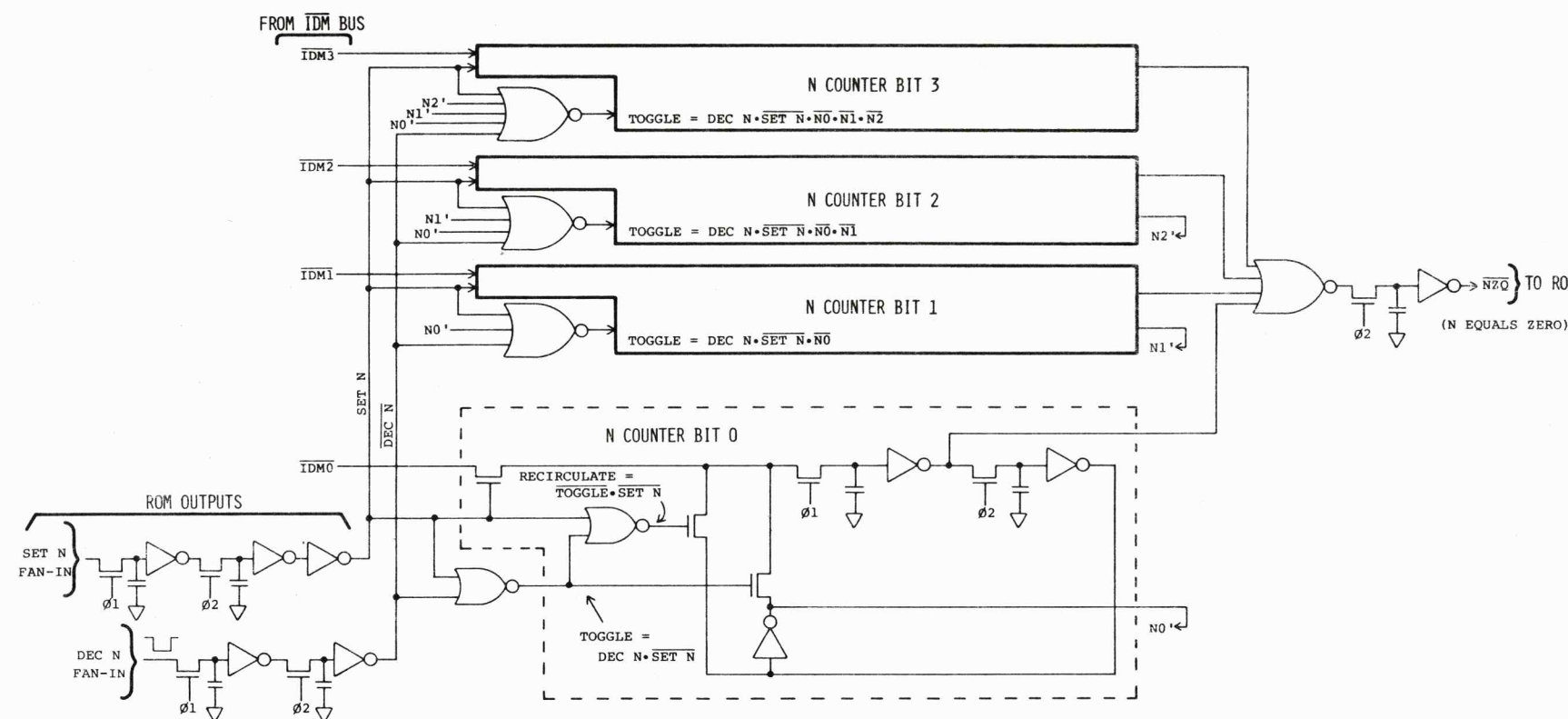


FIG 3-4

OVERVIEW OF THE CONTROL ROM

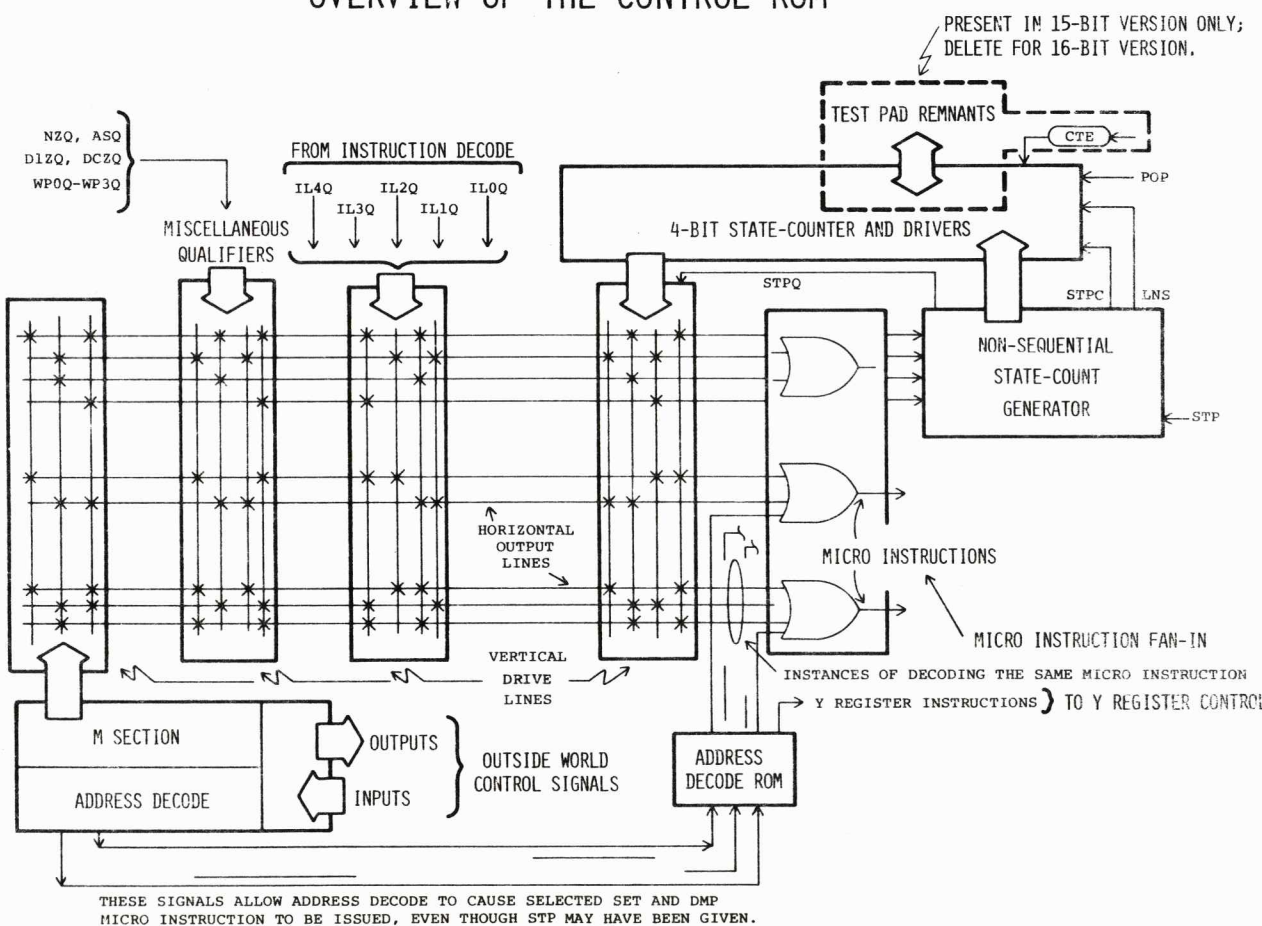


FIG 4-1

SECTION 3 (CONTINUED)

Figure 3-4 shows the details of the N Counter and of the micro-instructions SET N and DEC N. The decrementor is a standard binary decrementor. Observe the means used to generate NZQ. The phase two transfer gate ensures that NZQ is stable during phase one.

SECTION 4

Figure 4-1 is an overview of the Control ROM in the EMC. It is very similar to the ROM in the BPC.

The ROM has a 4-bit State-Counter whose operation is suspended with STP when a Bus Grant occurs. This State-Counter has a non-binary native state-count sequence. When branching considerations require that the native state-count sequence be overridden, a next-state micro-instruction is decoded. This causes the Non-Sequential State-Count Generator to issue Load Next State (LNS) and to encode and load the next state-count into the State-Counter. The newly supplied state-count, instead of the natural successor, will then be the next state.

The qualifier groups associated with the ROM are similar in arrangement and purpose to those of the BPC. However, one important difference exists in connection with Address Decode. It, in co-operation with the M-Section, supplies inputs to a separate ROM whose outputs are sets and dumps used to respond to memory cycles directed to registers within the EMC. The outputs of this separate ROM are not disabled by STP. Except for the Y register instructions, the outputs from the Address Decode ROM share the regular fan-in mechanism of the main Control ROM. That the Y register micro-instructions do not is due primarily to layout reasons. The distance between the Address Decode ROM and the Y registers is relatively short, while the fan-in mechanism of the main ROM was located a considerable distance from the Y registers.

The primary difference between the 15 and 16-bit versions involves the deletion in the 16-bit version of the test signal CTE (Counter Test

CONTROL ROM
OVERVIEW

N COUNTER

INSTRUCTION
LATCH

EMC INSTRUCTION
BIT PATTERNS

DETAILS OF THE ROM STATE-COUNTER

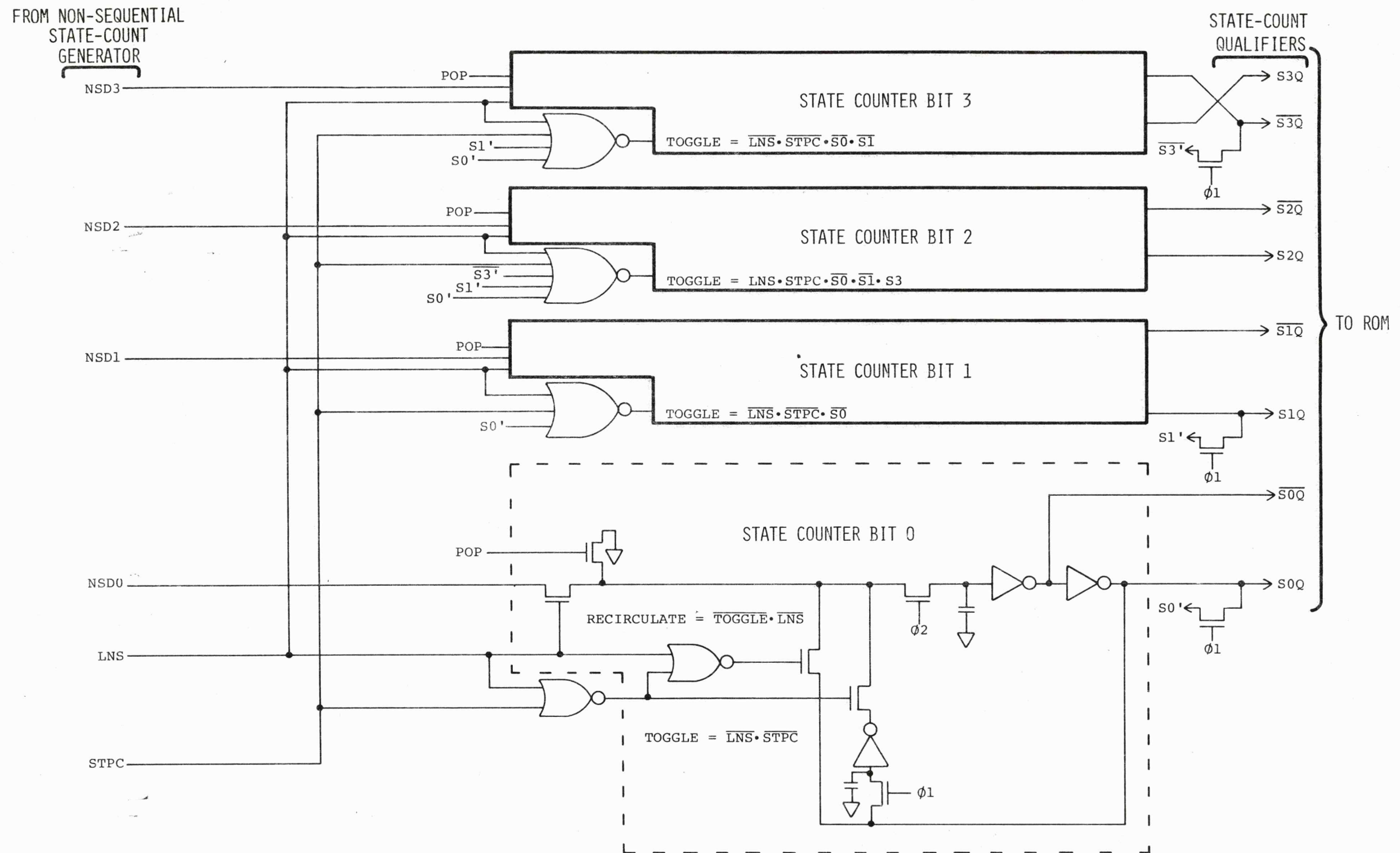


FIG 4-2-I-S

SECTION 4 (CONTINUED)

Enable) and some test pads. This was a developmental mechanism used to ensure proper operation of the State-Counter and as an aid in debugging the mechanism generally. It has no operational function and was removed from the 16-bit version to help provide room for the POP Synchronizer.

Figures 4-2-I-S and -F illustrate the details of the actual State-Counter. Observe how POP pre-sets each bit to zero.

In the absence of LNS and STPC (which is a derivative of STP) the State-Counter increments in a manner controlled by the gating shown. Each

bit either toggles or recirculates, based on the inputs to its own individual gating. The effect of LNS is to disable both the toggle and recirculate paths, while loading the state-count latch with a value determined by the Non-Sequential State-Count Generator.

The effect of STPC is to disable the toggle path and force the recirculate path. It does not do all of this directly; these are indirect results. STPC does directly disable the toggle feedback path. The absence of LNS is also required. This is achieved by having the presence of

DETAILS OF THE ROM STATE-COUNTER

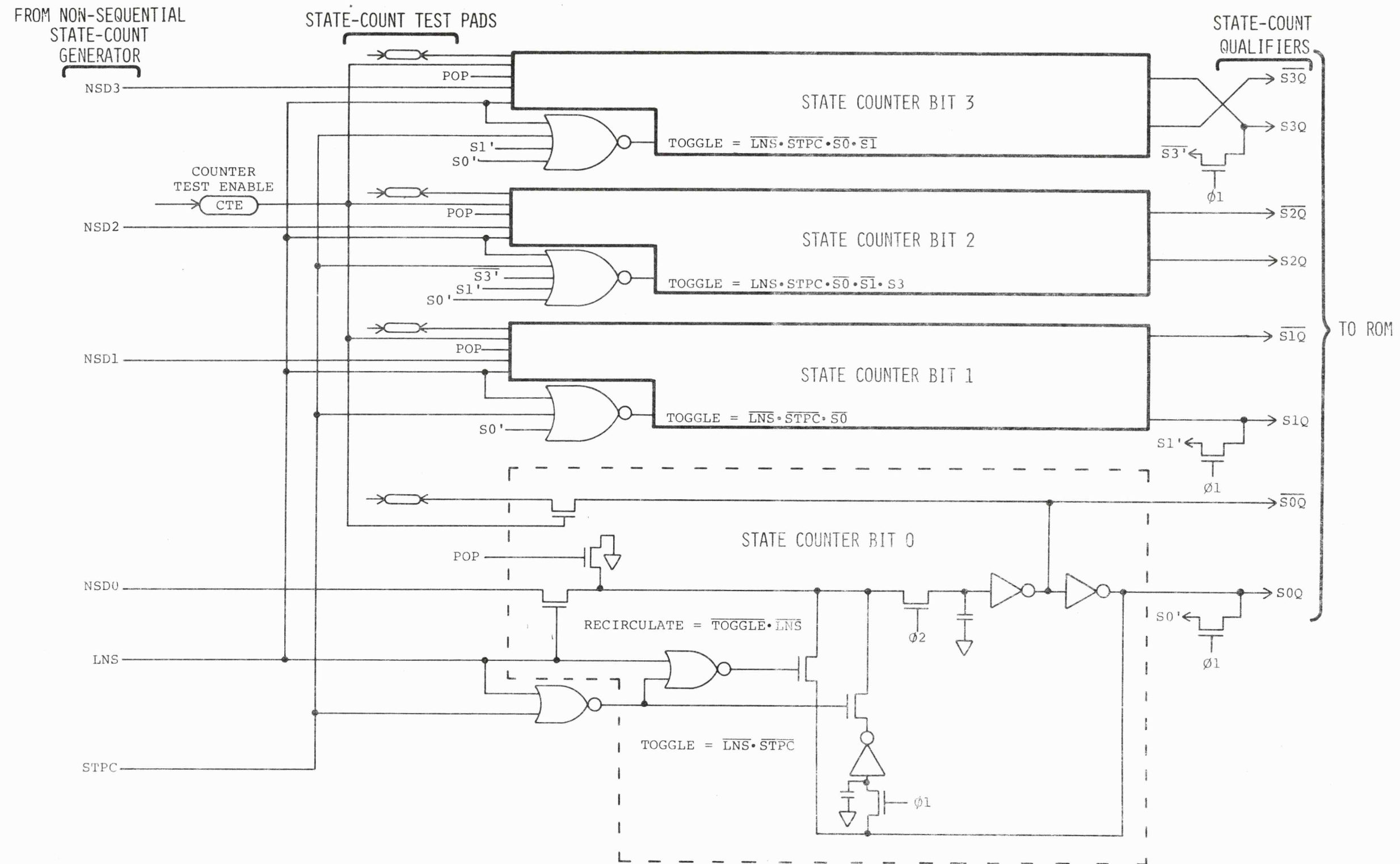


FIG 4-2-1-F

SECTION 4 (CONTINUED)

STP produce the absence of LNS. It is the absence of LNS, coupled with the absence of toggle, that enables the recirculate feedback path.

The difference between the 15 and 16-bit version concerns only the absence of the Counter Test Enable function and the test pad remnants.



ROM STATE-COUNTER



ROM STATE-COUNTER

ROM STATE-COUNT PATTERNS

AS1 CHART STATE #	ROM STATE-COUNTER OUTPUTS			
	S3Q	S2Q	S1Q	S0Q
0	0	0	0	0
1	1	0	1	1
2	1	0	1	0
3	1	0	0	1
4	1	0	0	0
5	0	1	1	1
6	0	1	1	0
7	0	1	0	1
8	0	1	0	0
9	1	1	1	1
10	1	1	1	0
11	1	1	0	1
12	1	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
0	0	0	0	0

FIG 4-2-2

SECTION 4 (CONTINUED)

Figure 4-2-2 shows the ROM state-count pattern produced by the incrementing sequence employed in the EMC's State-Counter.

Figure 4-3 illustrates the means used to decode the micro-instructions from the ROM. The technique used is identical to that used in the BPC. Refer to the description of the BPC for more information.

DETAILS OF ROM OUTPUT DECODING

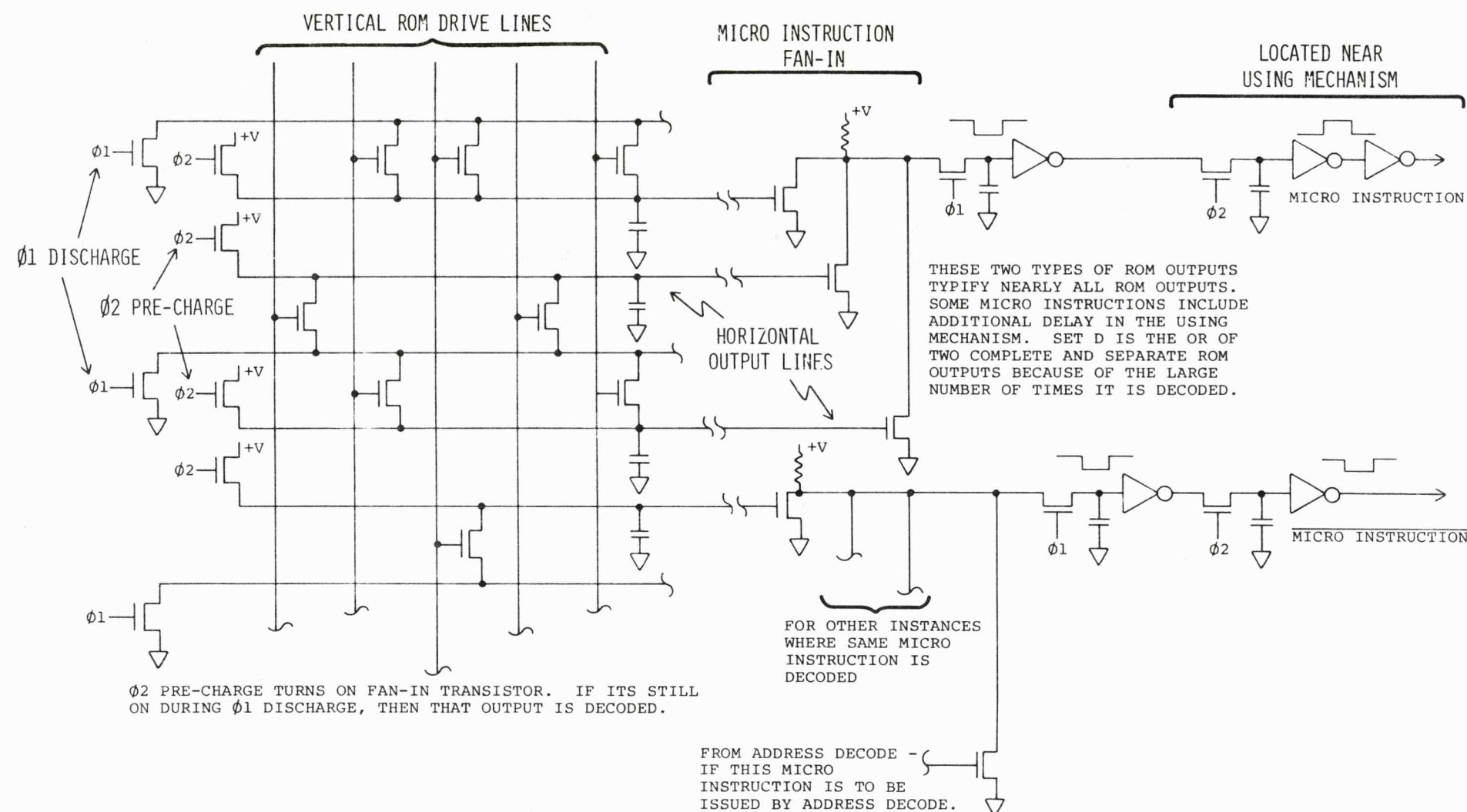


FIG 4-3

OVERVIEW OF THE NON-SEQUENTIAL STATE-COUNT GENERATOR

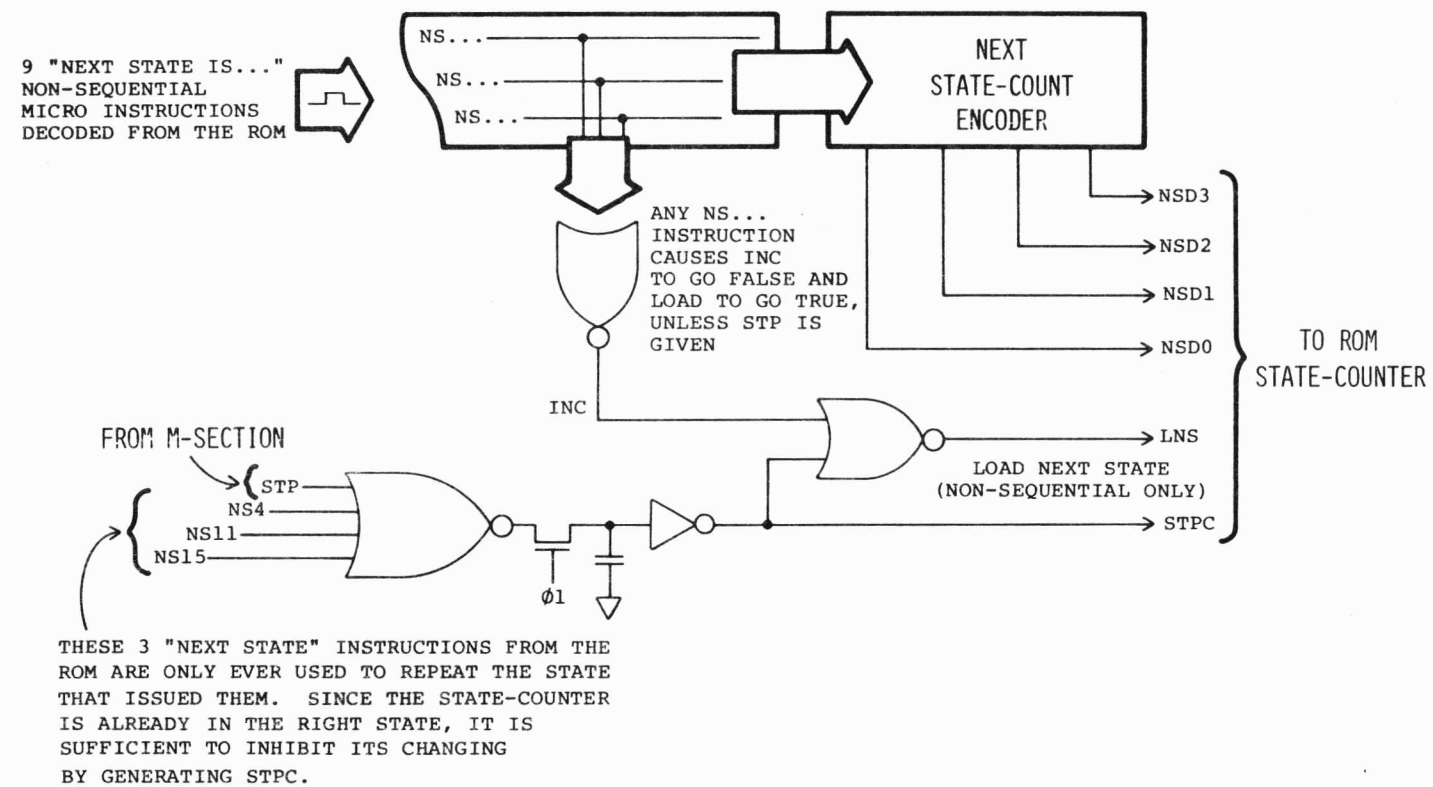


FIG 4-4-1

SECTION 4 (CONTINUED)

Figure 4-4-1 is an overview of the Non-Sequential State-Count Generator. There are nine non-sequential next-state micro-instructions. Each is used to produce a particular pattern in the Next State-Count Encoder. The occurrence of any of these non-sequential micro-instructions causes INC to go false, which in turn causes LNS to go true.

There is one bit of funny business associated with the Non-Sequential State-Count Generator. There are three additional non-

sequential micro-instructions (which brings the total to twelve) which have special properties. These are NS4, NS11 and NS15. These three are only ever used to repeat the state that issued them. A slight savings in the component count was achieved by arranging that these micro-instructions issue STPC. The effect is to prevent the state-counter from changing, rather than reloading it with it's old value.

Observe that STP causes LNS to go false.

DETAILS OF THE NEXT STATE-COUNT ENCODER AND THE GENERATION OF INC

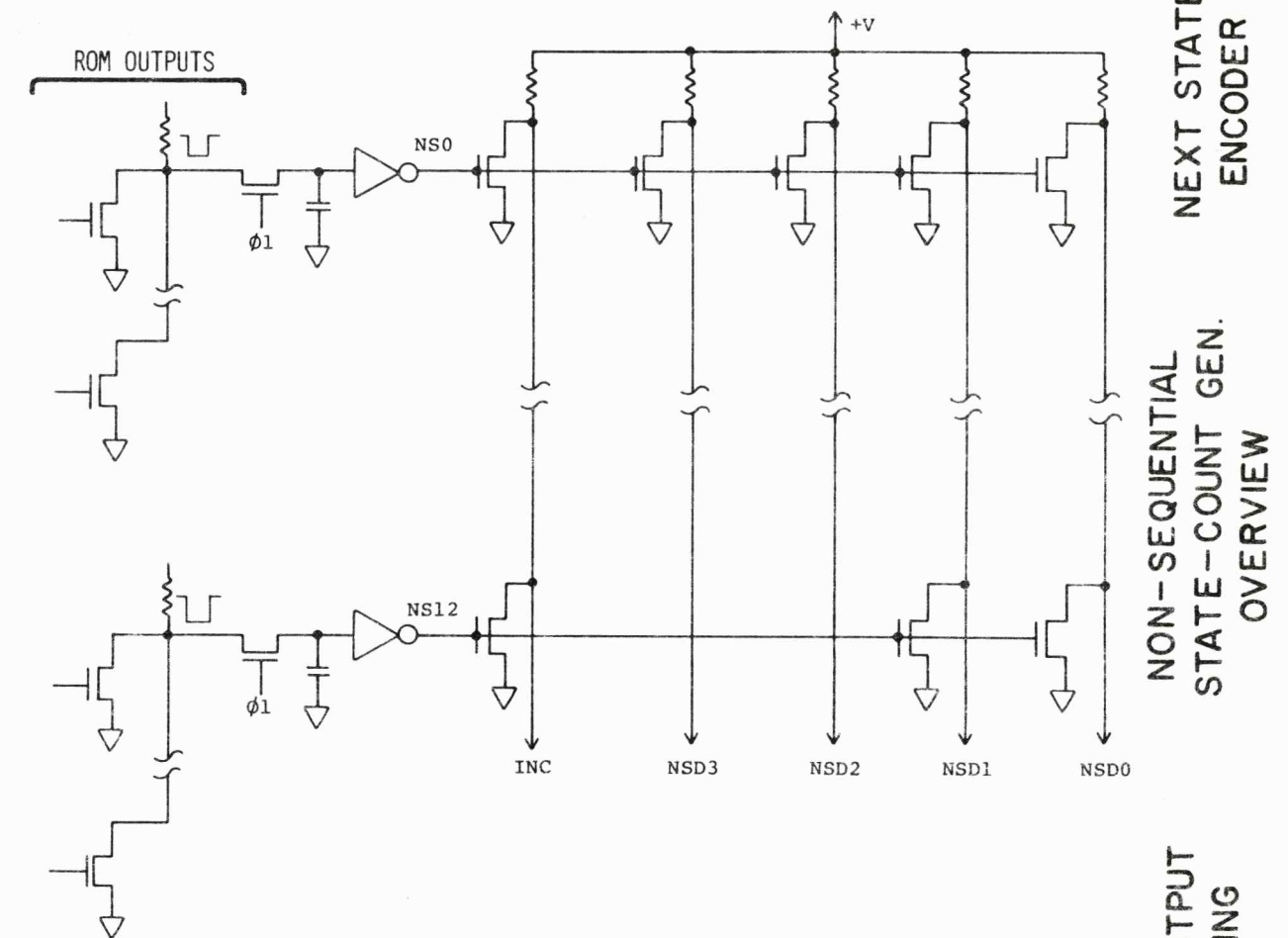


FIG 4-4-2

Figure 4-4-2 illustrates the details of the Next State-Count Encoder and of the generation of INC. It shows how each "NS" micro-instruction is capable of putting a particular pattern on the NSD lines, as well as causing INC to simultaneously go false.

ROM STATE-COUNT GENERATOR OVERVIEW

ROM STATE-COUNT PATTERN

OVERVIEW OF THE AE REGISTER

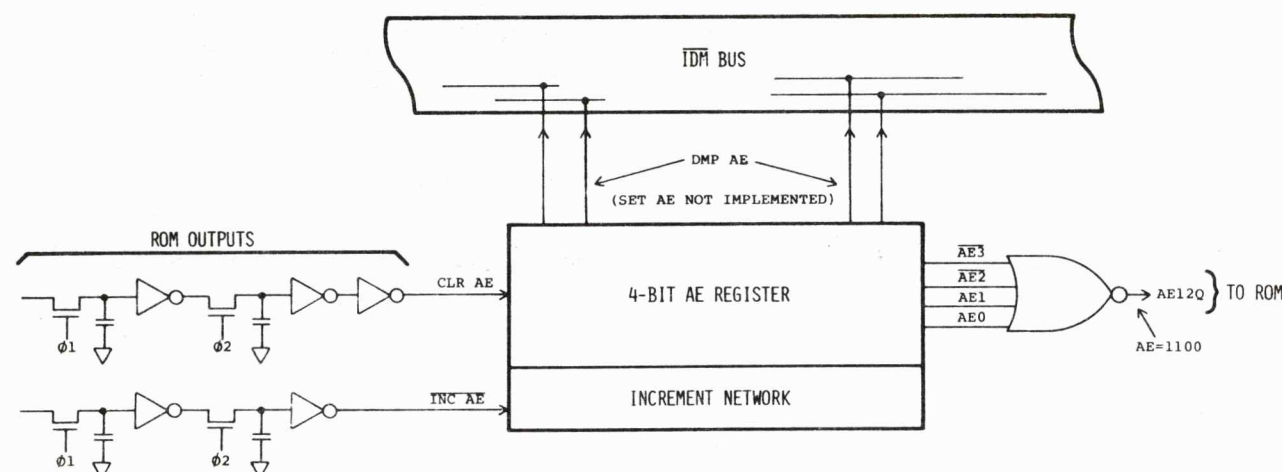


FIG 5-1

SECTION 5

Figure 5-1 is an overview of the AE register and its associated circuitry. The AE register is a 4-bit register connected to the least four bits of the IDA Bus. Its purpose is to assist in the execution of certain arithmetic instructions; in particular, FMP, FDV and NRM.

AE is always used as a counter. To this end, it can be cleared, incremented and dumped, but not set. During the execution of FMP and FDV machine-instructions AE is used to record the number of times overflow occurs during BCD additions or subtractions. During the execution of the NRM machine-instruction AE is used to count the number of shifts that are performed. Since it doesn't make sense to do more than eleven such shifts the qualifier AE12Q has been provided to indicate that the maximum necessary number of shifts has been performed.

Figure 5-2 shows the details of the circuitry of the AE register. There is nothing particularly remarkable about this circuitry.

SECTION 6

Figure 6-1 is an overview of Shift Extend (SE) and of the various X registers, including the SIZ Latch and Shift Control. The primary purpose of the SE register is to assist in shift operations. Shifted digits pass through it, and at the conclusion of the shift operation the last digit shifted out is retained, and is thereby available for inspection. The primary purpose of the X register is to manipulate the value of AR1 during mantissa shift or BCD arithmetic machine-instructions. (The Y registers are AR2. The X registers are not the same as AR1, however. AR1 is located in memory outside the EMC and is read into the X registers, manipulated, and the final result placed back into AR1.) The X registers will be used to manipulate only the mantissa portion of AR1 or AR2. Accordingly, there is no X0 to correspond to the exponent word.

The X registers are used to provide two types of operations. The first of these has to do with actual BCD arithmetic. This involves the need to put one of the three registers onto the IX Bus, from which it goes to the Adder. Only one of the X

DETAILS OF THE AE REGISTER

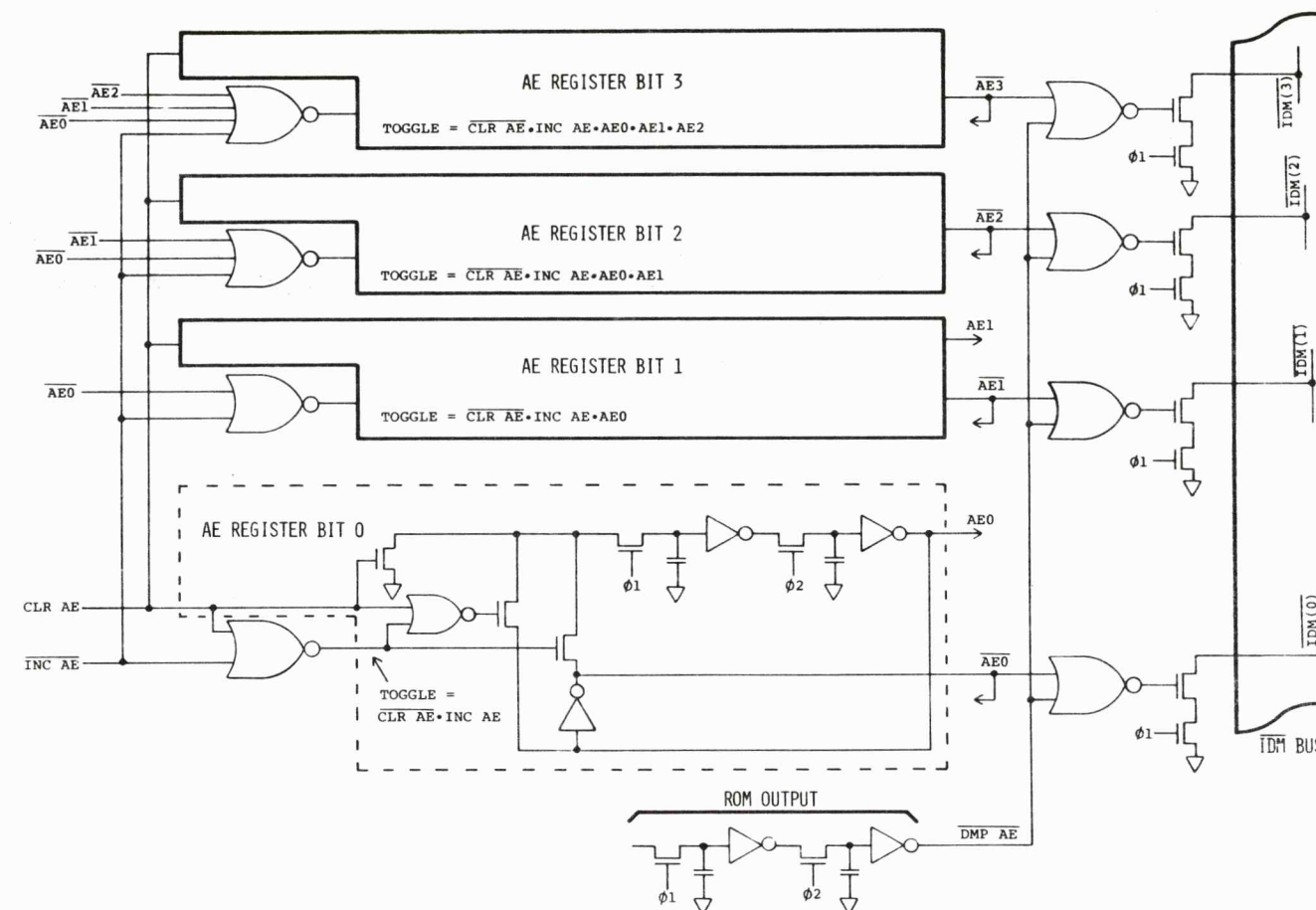


FIG 5-2

registers at a time is placed on the IX Bus. The selection is determined by the Word Pointer. The selected path is essentially a direct connection and does not require any additional micro-instructions to be effective. The second major operation is shifting. The combination SE-X1-X2-X3-SE makes a giant bi-directional shift register. Shift operations are conducted one bit at a time, but always in groups of four bits as a minimum unit. Various shift signals are used, as described below.

As previously mentioned, the value of the Word Pointer selects which X register's direct connection to the IX Bus is enabled. It does more than this, however. It also selects which X register is to be set or dumped upon receipt of a SET X or DMP X. The function just described is one of the services provided by X Register Control. Note that SET and DMP instructions that explicitly

reference a particular X register are also available.

The purpose of the ZIX circuit is to provide a means to force the IX Bus to be zero, regardless of the contents of the selected X register.

Shifting operations are produced by the micro-instruction Shift Register Enable (SRE) in conjunction with the asynchronous control lines. SRE is gated with various control lines to produce one or more of the following signals:

- SR Shift Right; used for X2, X3 & SE.
- SR1 Shift X1 Right; used only for X1.
- SL Shift Left; goes to all X's & SE.

Each time one of these signals is issued the affected elements shift one bit in the indicated direction. The reason for shifting X1 separately from X2, etc., has to do with the Booth's Multiply algorithm, and is explained at a later time.

OVERVIEW OF THE X AND SE REGISTERS

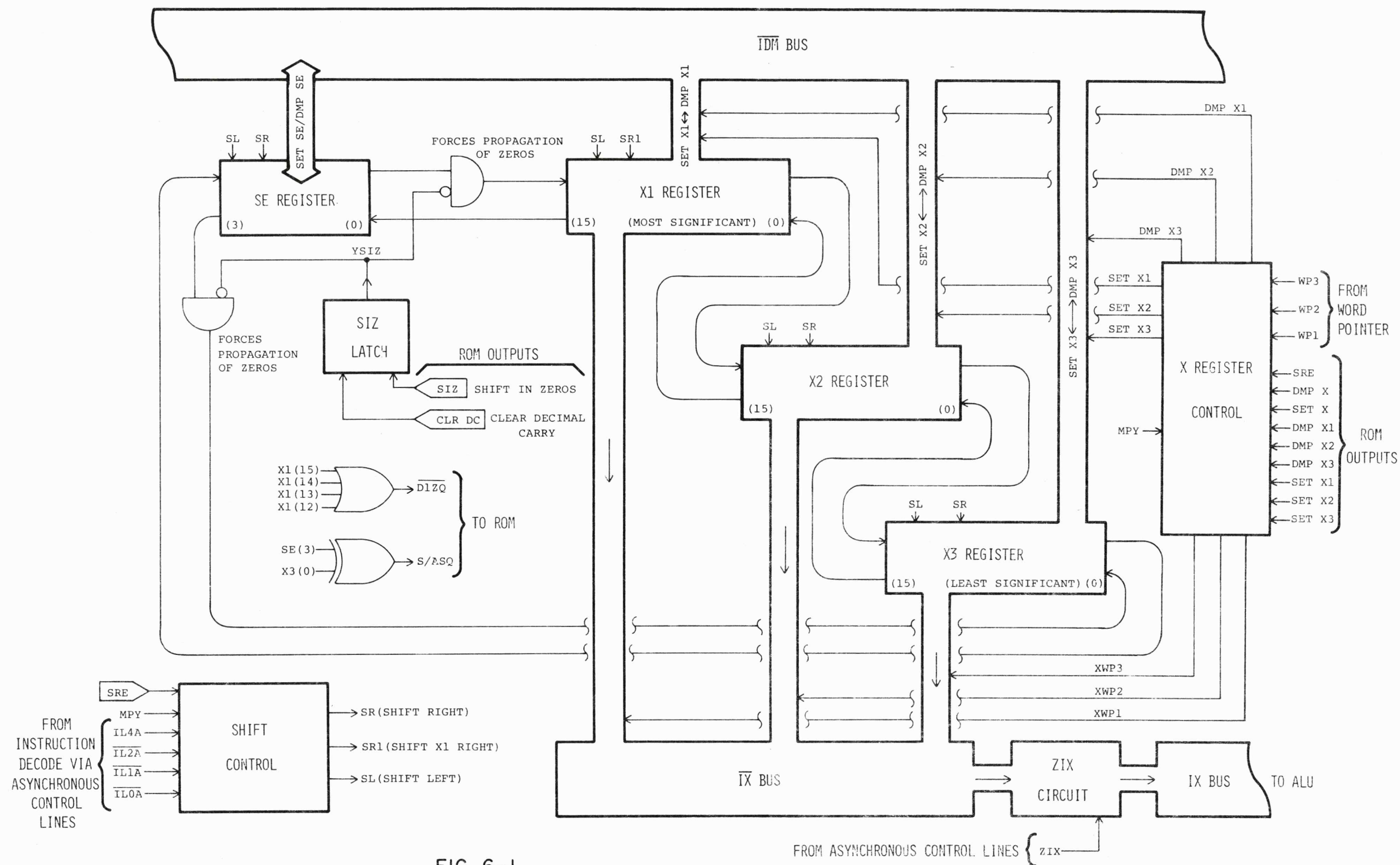


FIG 6-1

SECTION 6 (CONTINUED)

The shifting mechanism provides the means to shift zeros into the X registers in two ways. First, zeros may be right-shifted into the most significant portion of X1, overriding the wrap around from X3. Second, zeros may be left-shifted into the

bottom of X3, overriding the wrap around from the top of X1. The shifting in of zeros is controlled by the SIZ Latch. To shift in zeros the latch is set by issuing SIZ. The latch will remain set until CLR DC is issued. Two qualifiers are shown on

the drawing. These are DIZQ and S/ASQ. DIZQ means that the most significant BCD digit of the X registers is zero. This is a condition of importance to the flow charting corresponding to several machine-instructions. S/ASQ has to do with Booth's algorithm for

binary multiplication. All of the stuff concerning Booth's algorithm is explained at the time the flow chart for that operation is presented.

X REG, SE REG
AND SIZ LATCH
OVERVIEW

AE REGISTER

AE REGISTER
OVERVIEW

DETAILS OF X REGISTER CONTROL

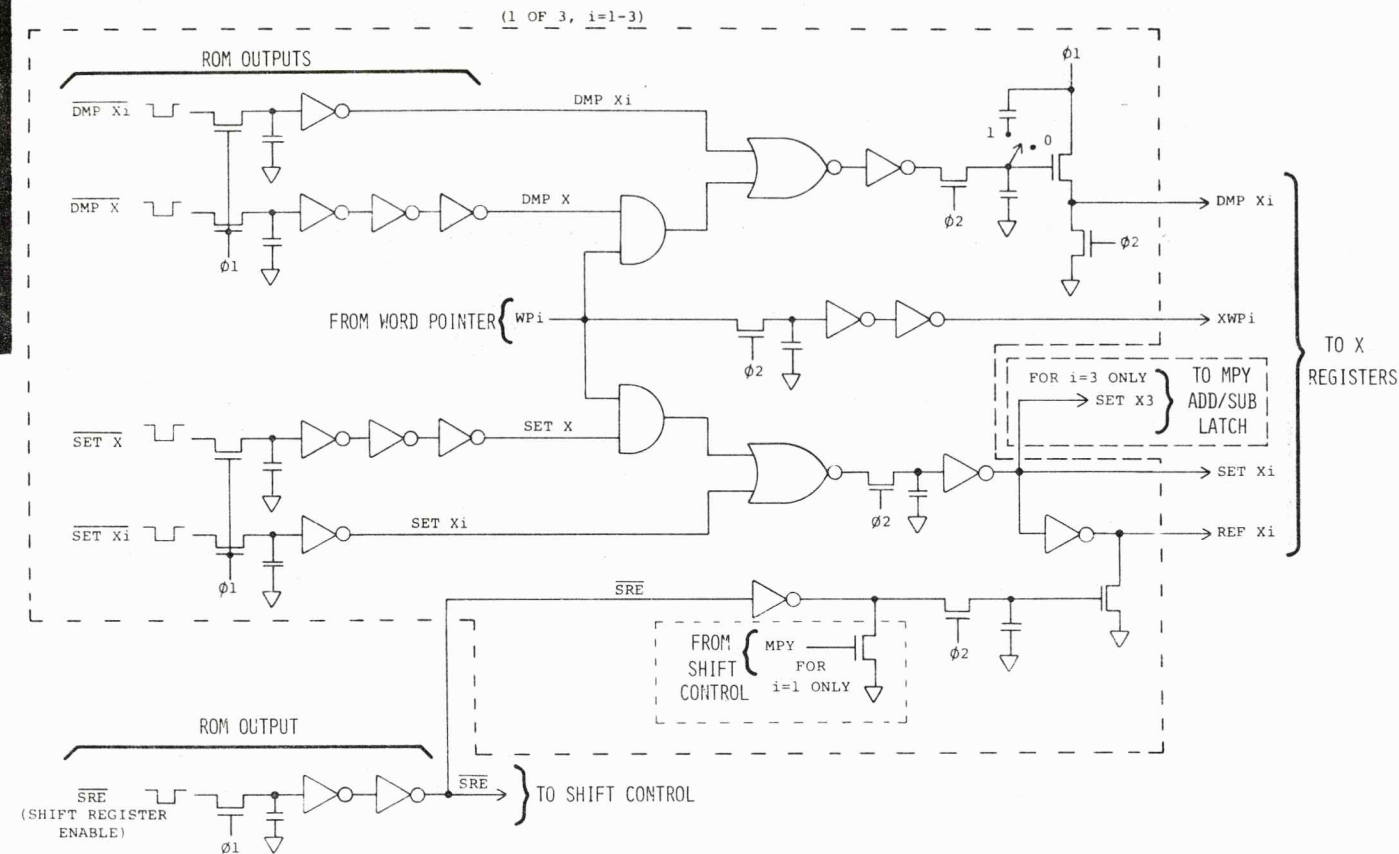


FIG 6-2

SECTION 6 (CONTINUED)

Figure 6-2 illustrates the details of X Register Control. Primarily, it illustrates how the general SET and DMP X micro-instructions are merged with their explicit counterparts according to the value of the Word Pointer. It also illustrates how SRE is affected by the MPY condition (which is the Booth's algorithm thing). Normally, SRE disables the refresh signal for each register affected by the shift. The effect on SRE is to preserve the refresh of X₁ during the Booth's algorithm multiply.

DETAILS OF SHIFT CONTROL

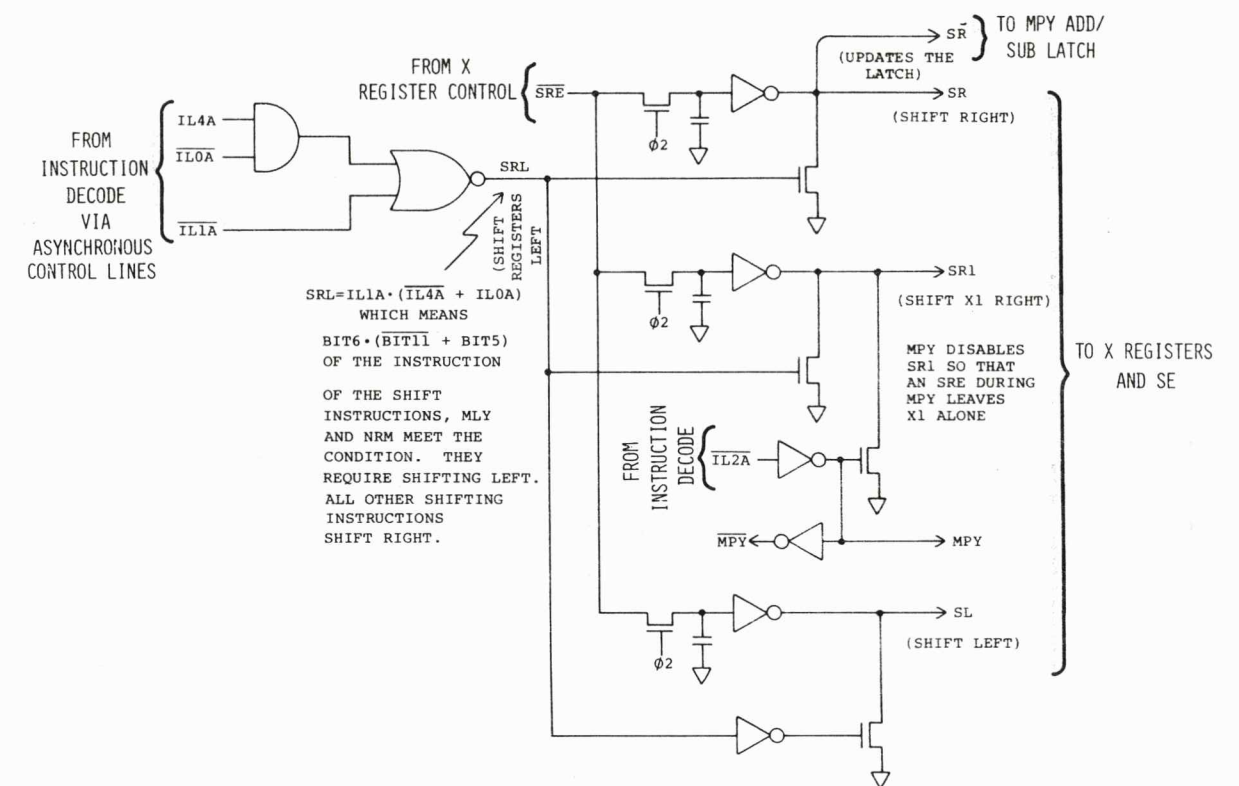


FIG 6-3

Figure 6-3 illustrates the details of Shift Control. Basically, this determines which combination of shift-left and shift-right control signals to issue when SRE occurs. Most of the funny business here concerns the binary multiply, which is described later.

DETAILS OF THE X1 REGISTER

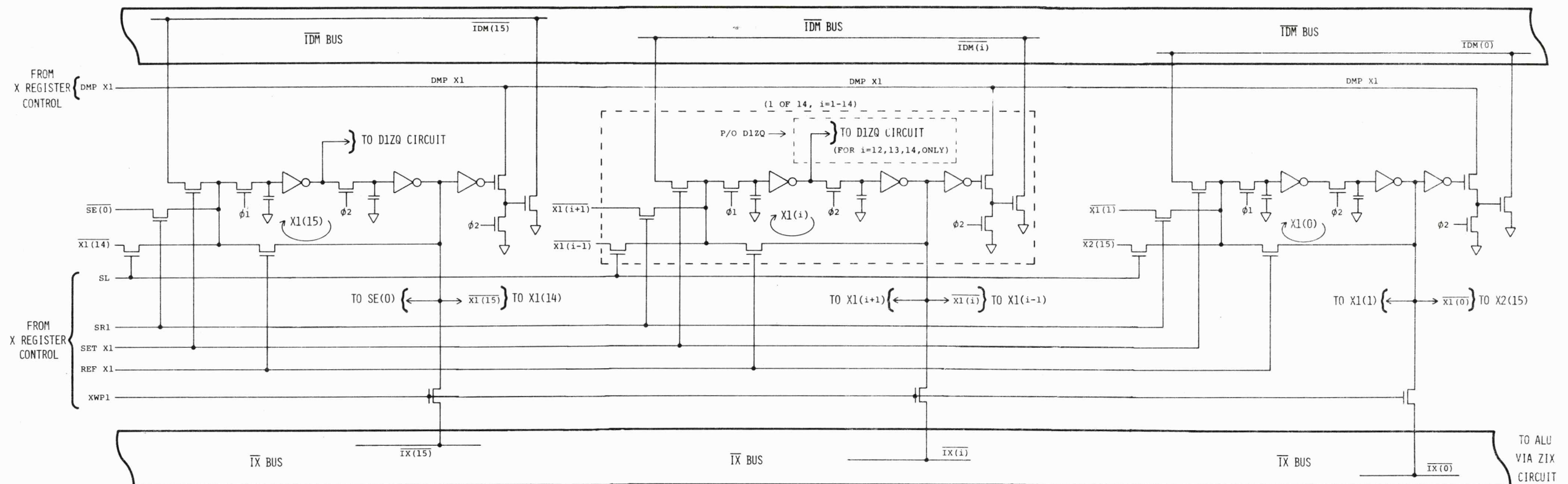


FIG 6-4-1

SECTION 6 (CONTINUED)

Figure 6-4-1 illustrates the details of the X1 register. As a register, it is rather straightforward. Observe the connections for shifting left and right. Also notice the direct connection to the IX Bus whenever XWP1 is true. And lastly, notice the tapped connections for D1ZQ on bits 12-15.

Figure 6-4-2 illustrates the details of the X2 register. Observe the direct connection established between the X2 register and the IX Bus when XWP2 is true. There is more funny business connected with MPY; observe that MPY breaks the shift-right connection between X1 and X2. This is shown in the drawing for X2(15). Also notice that MPY causes X2(15) to shift into itself during a right shift. This establishes the necessary binary arithmetic right shift for X2 and X3 during MPY operation.

Figure 6-4-3 illustrates the details of the X3 register. The basic form of this register is similar to X1 and X2. In fact, it is the simplest of the bunch. Observe that bit zero of this register is tapped off and sent to various circuits that are interested in it.

XI REGISTER

SHIFT CONTROL

X REGISTER CONTROL

DETAILS OF THE X2 REGISTER

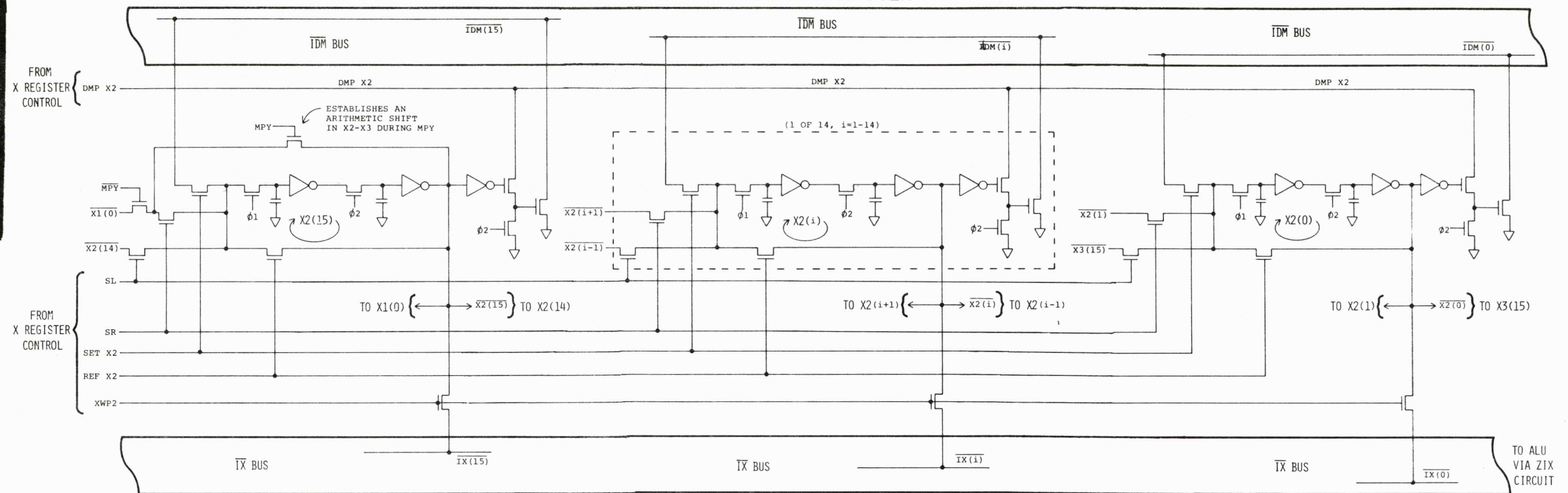


FIG 6-4-2

DETAILS OF THE X3 REGISTER

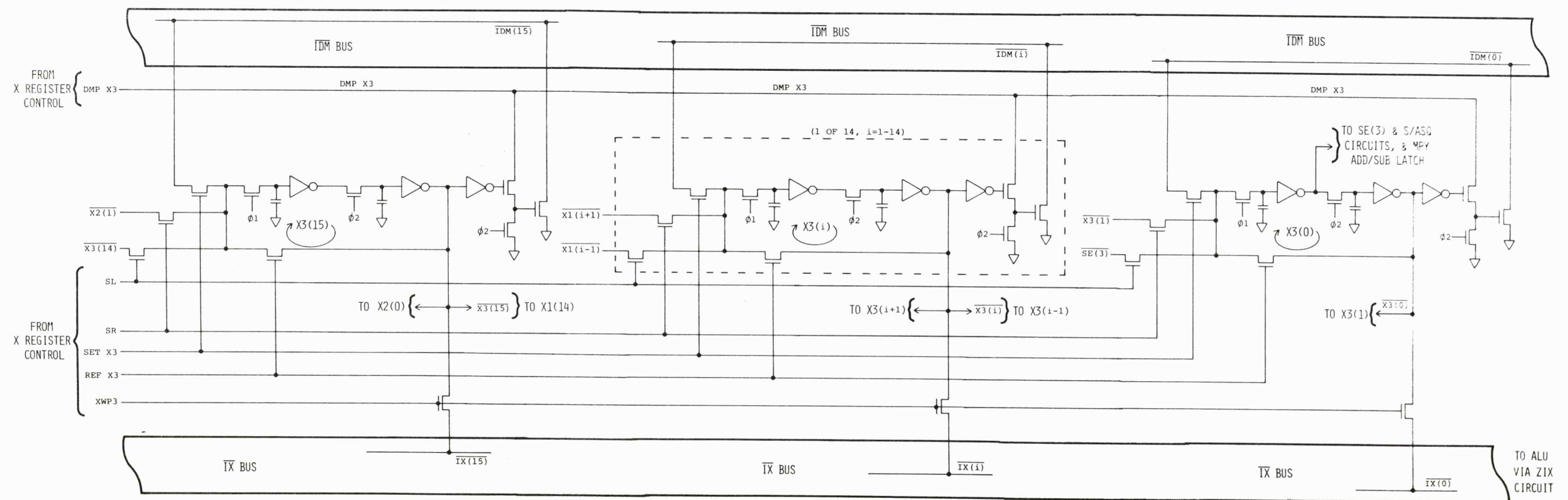


FIG 6-4-3

DETAILS OF THE SE REGISTER

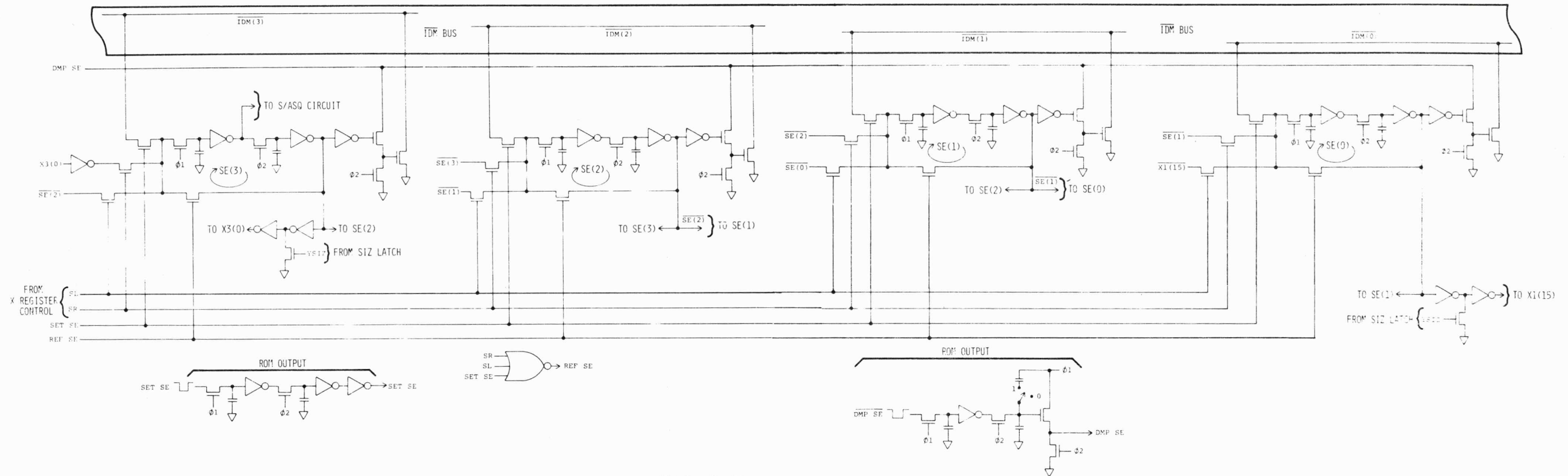


FIG 6-5

SECTION 6 (CONTINUED)

Figure 6-5 illustrates the details of the SE register. Observe how the shift-in-zero mechanism is appended to bit 3 and bit 0.

Figure 6-6 shows the details of the SIZ Latch. This latch is set by the micro-instruction SIZ, and cleared with the micro-instruction CLR DC. It produces the signal YSIZ, which is used in the SE register to propagate zeros during shifting.

Figure 6-7 shows the details of the ZIX circuit. This circuit is used to force the IX Bus to all zeros during execution of certain machine-instructions.

Figure 6-8 shows the details of the DIZQ circuit.

Figure 6-9 shows the details of the S/ASQ circuit. The usefulness of this circuit will be explained during the explanation of Booth's algorithm.

SE REGISTER

X3 REGISTER

X2 REGISTER

DETAILS OF THE SIZ LATCH

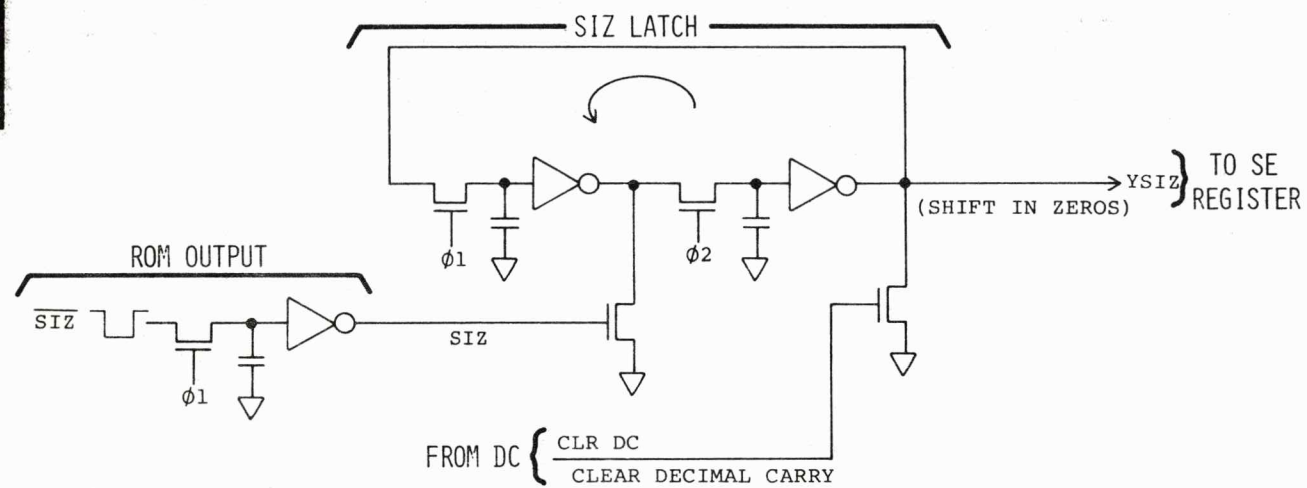


FIG 6-6

DETAILS OF THE DIZQ CIRCUIT

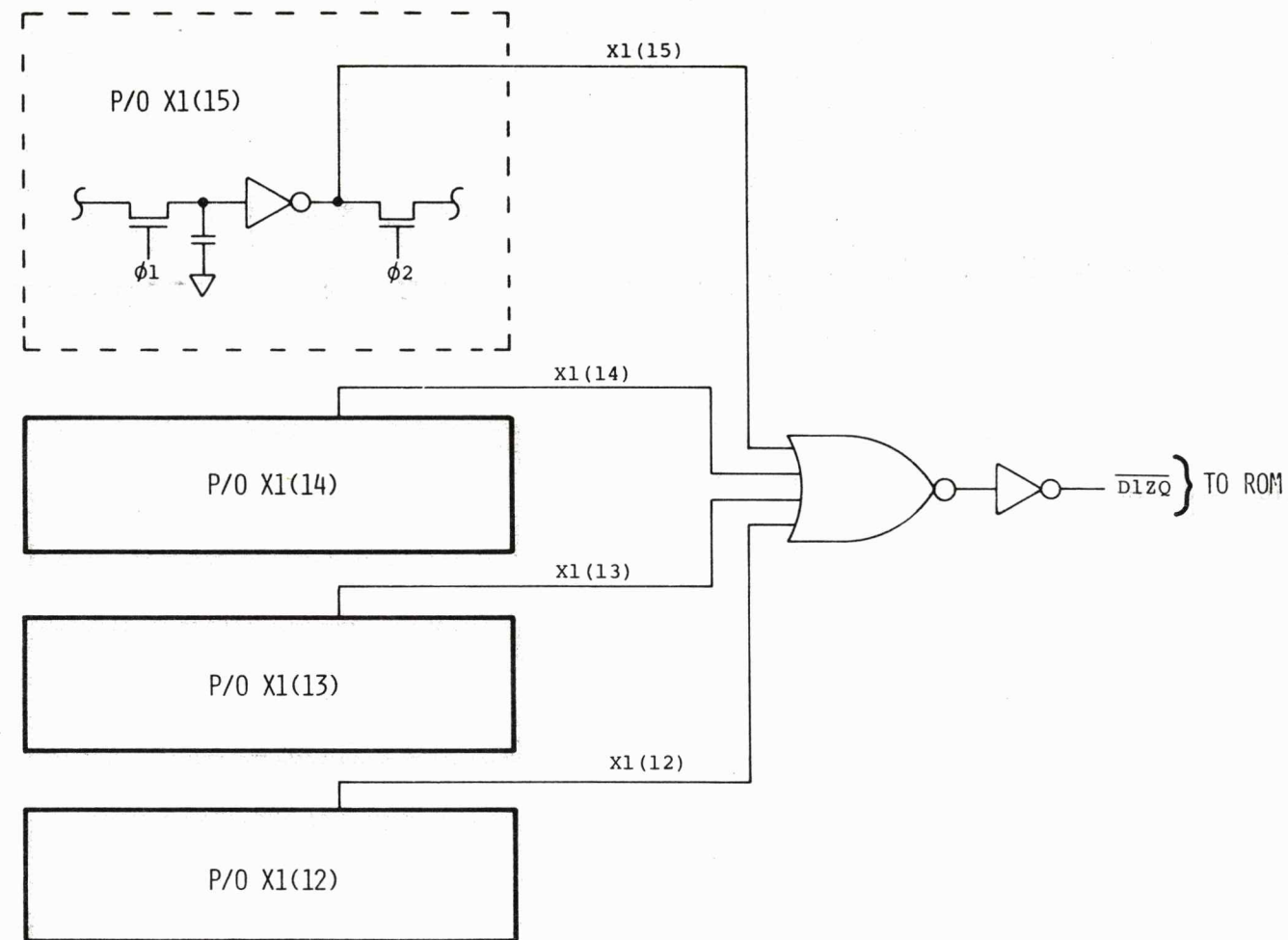


FIG 6-8

DETAILS OF THE ZIX CIRCUIT

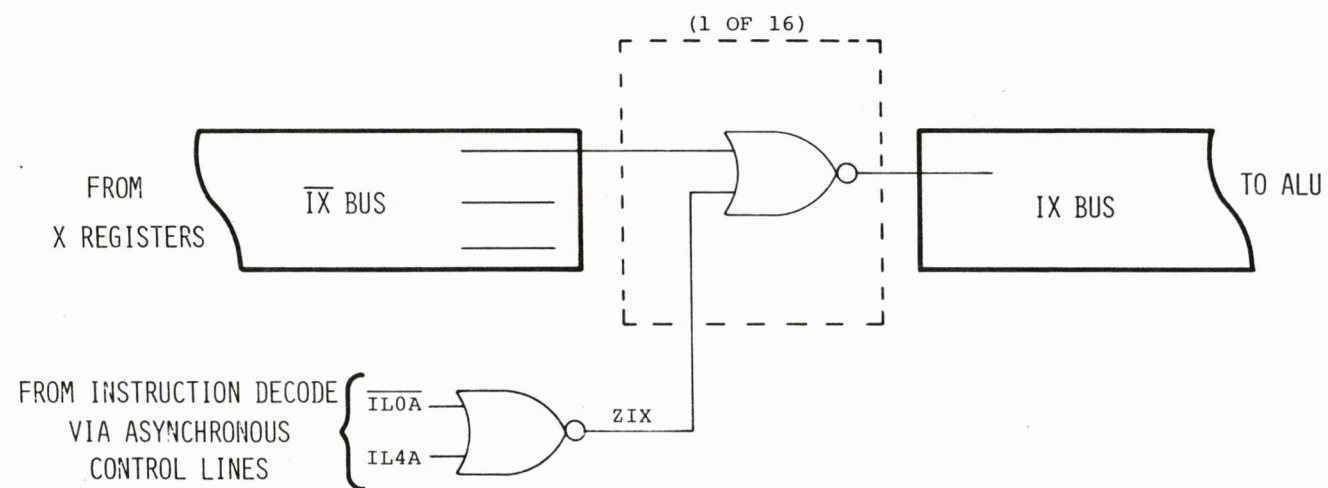


FIG 6-7

DETAILS OF THE S/ASQ CIRCUIT

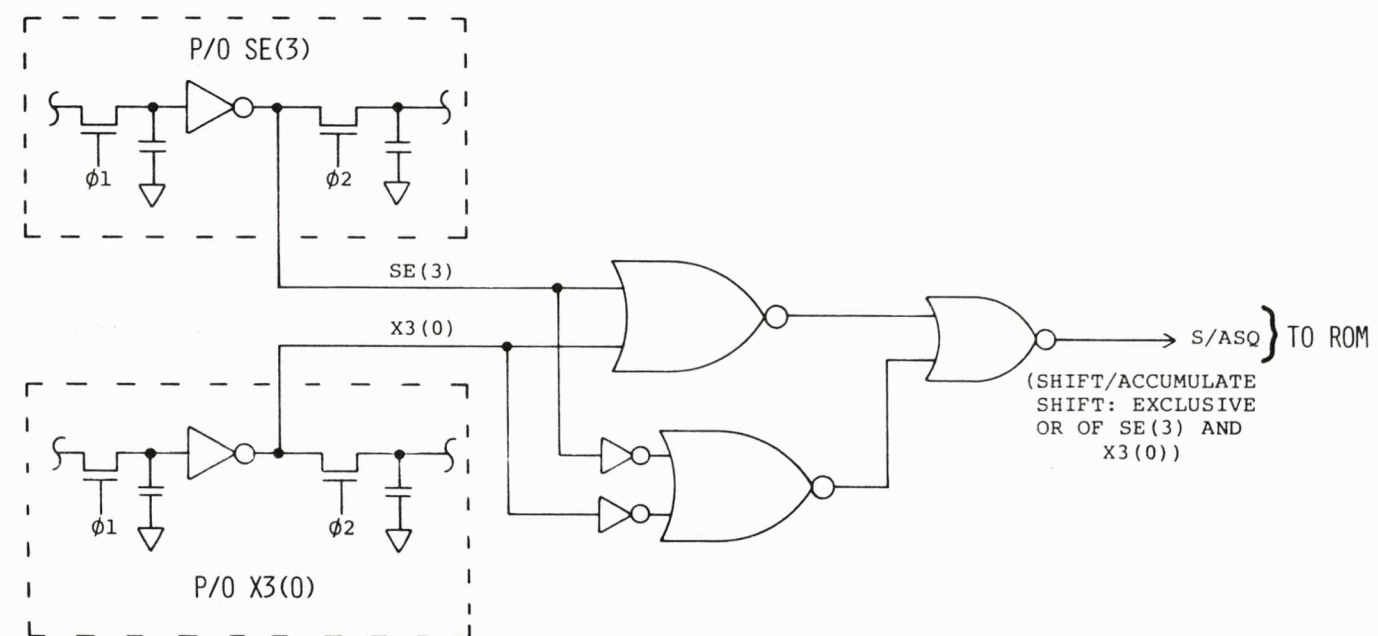


FIG 6-9

OVERVIEW OF THE WORD POINTER SHIFT REGISTER AND WORD POINTER ENCODER

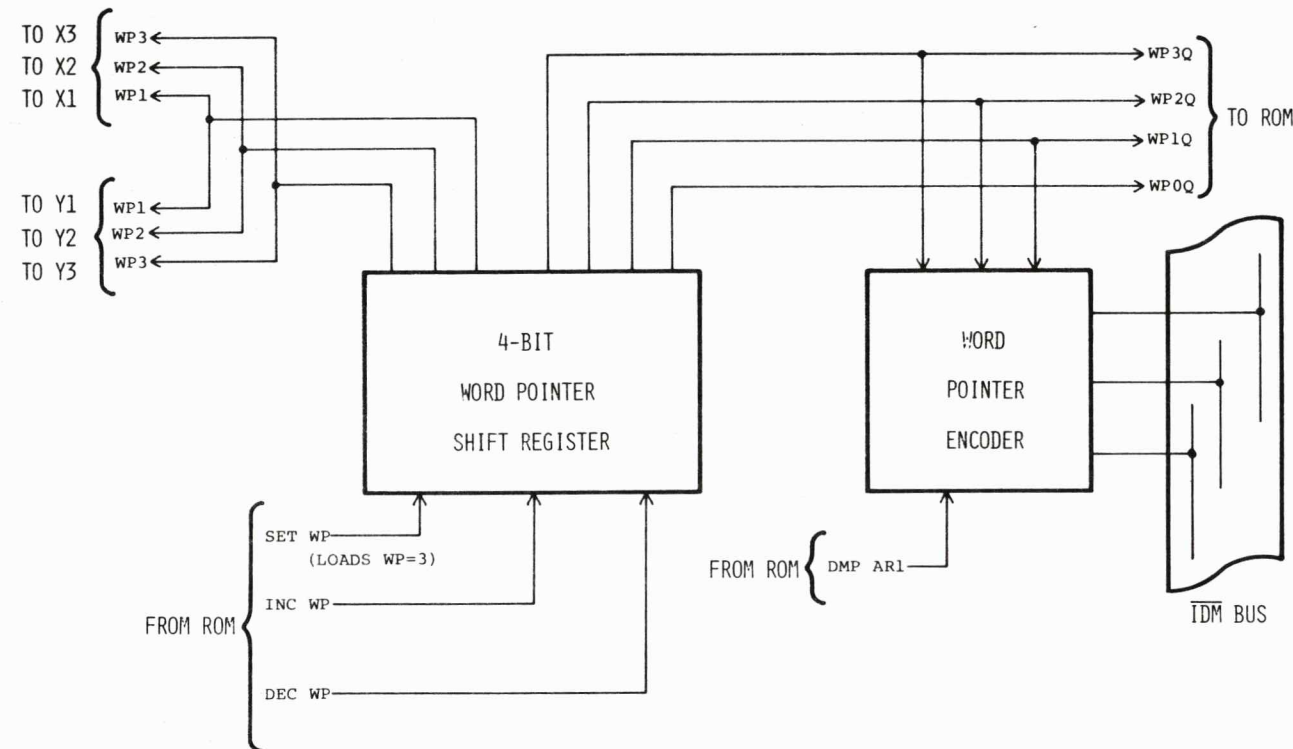


FIG 7-1

SECTION 7

Figure 7-1 is an overview of the Word Pointer Shift Register and of the Word Pointer Encoder. The purpose of the Word Pointer is to select which portions of the X and Y registers are to be dealt with by the Adder, or set or dumped in response to general set or dump instructions.

The Word Pointer is not an actual counter. It is a shift register which will have one bit set. The position of the set bit indicates the count. The micro-instructions

INC WP and DEC WP each cause a shift of one bit position (although in opposite directions). The micro-instruction SET WP sets the Word Pointer to indicate a count of three.

The outputs of the Word Pointer go to four places. These are to the X and Y registers, to the Word Pointer Encoder, and to the ROM. The Word Pointer Encoder is used in generating the address of ARI. Recall that this is in read/write memory outside the EMC. Also recall that it is a four

DETAILS OF SET WP, REF WP, INC WP AND DEC WP

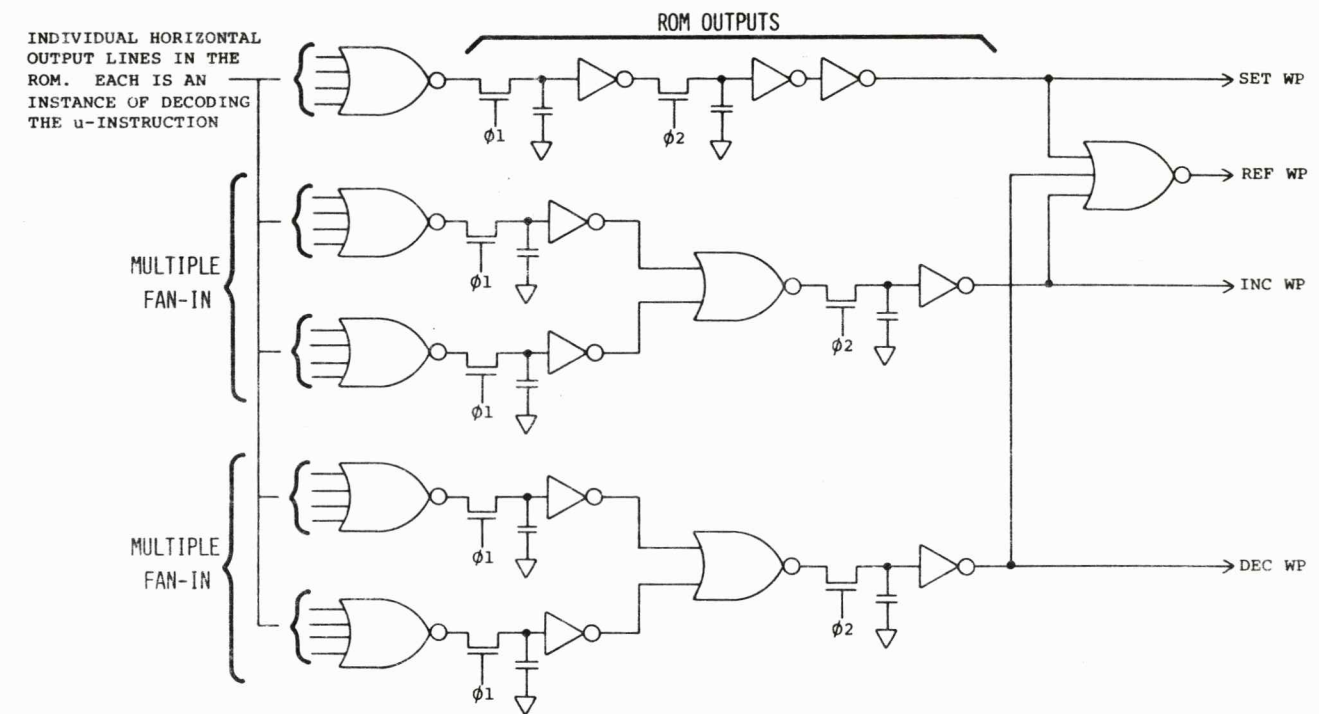


FIG 7-2

word quantity, and as such does not have a single address. ARI is represented by a group of four consecutive addresses. The micro-instruction DMP ARI provides the upper thirteen bits (for the 15-bit case) or fourteen bits (in the 16-bit case) of the address of ARI. The state of the Word Pointer selects which of the four segments of ARI is to be addressed. The Word Pointer Encoder trades the 1-of-4 representation used by the Word Pointer for a 2-bit binary re-

presentation, which is then concatenated with the upper thirteen or fourteen bits provided by DMP ARI.

Figure 7-2 shows the details of the SET WP, REF WP, INC WP and DEC WP micro-instructions. In the usual manner, the absence of setting, incrementing or decrementing the Word Pointer causes it to be refreshed.

WORD POINTER &
WORD POINTER ENCODER
OVERVIEW

DIZQ CIRCUIT
S/ASQ CIRCUIT

SIZ LATCH
ZIX CIRCUIT

DETAILS OF THE WORD POINTER SHIFT REGISTER

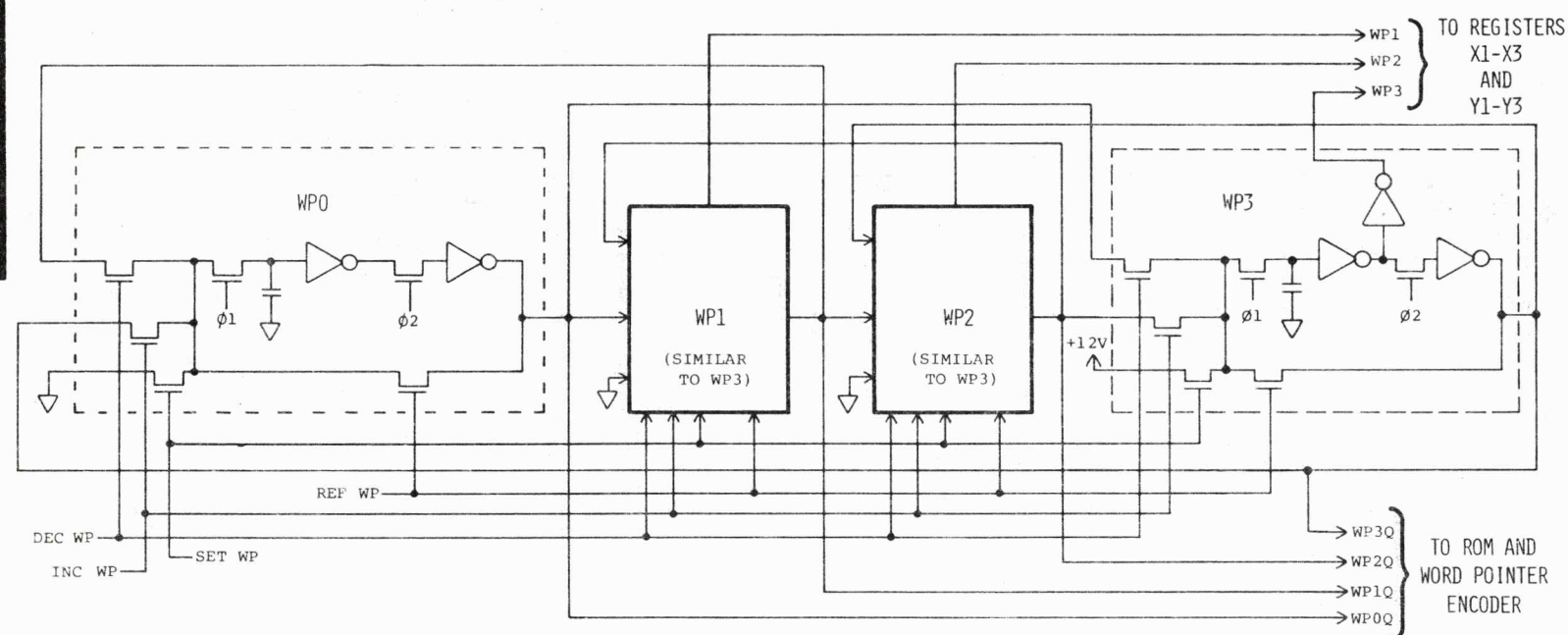
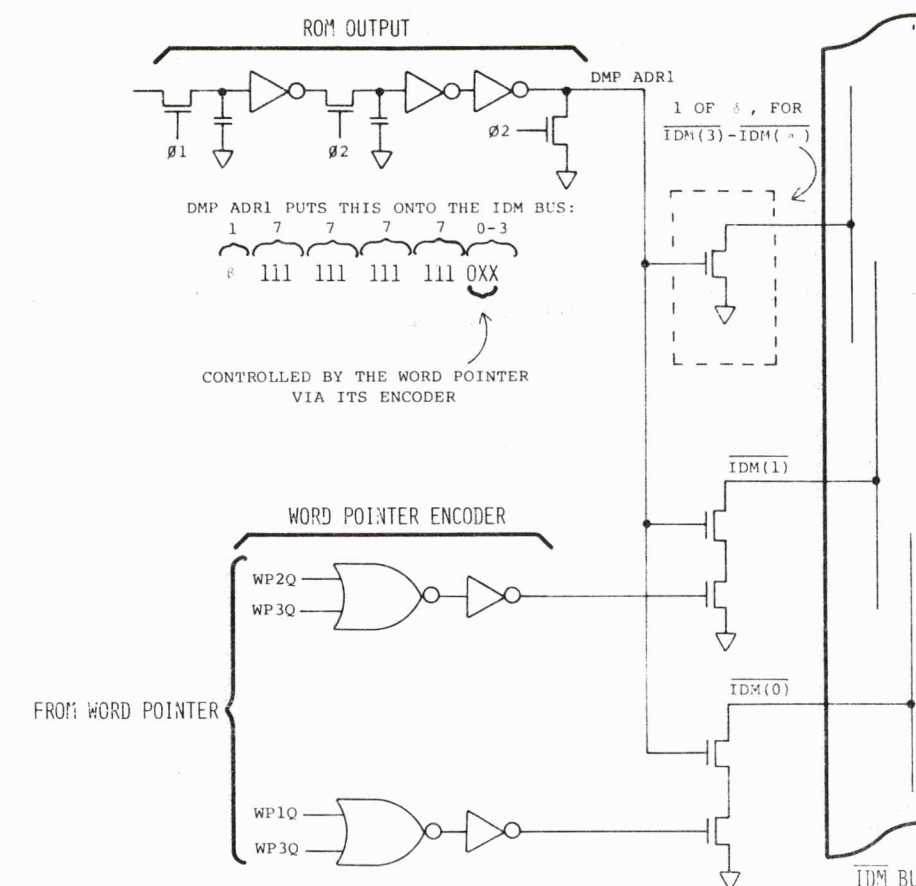


FIG 7-3

SECTION 7 (CONTINUED)

Figure 7-3 shows the details of Word Pointer proper. Note that the shift register is closed upon itself and is capable of wrapping around.

DETAILS OF DMP ADR1 AND THE WORD POINTER ENCODER



THIS PATTERN NOT USED →

WORD POINTER				IDM BUS	
WP3	WP2	WP1	WP0	IDM(1)	IDM(0)
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

- = 12 FOR 15-BIT VERSION AND 13 FOR 16-BIT VERSION
- = 0 FOR 15-BIT VERSION AND 1 FOR 16-BIT VERSION
- = 14 FOR 15-BIT VERSION AND 15 FOR 16-BIT VERSION

FIG 7-4

Figure 7-4 illustrates the details of Word Pointer Encoder. It is a straightforward case of binary encoding onto the two least significant bits of the IDM Bus. Observe how the micro-instruction DMP ADR1 puts the rest of the address for AR1 onto the IDM Bus.

The difference between the 15 and 16-bit versions concerns only the most significant address bit provided by DMP ADR1.

OVERVIEW OF THE Y REGISTERS AND Y REGISTER CONTROL

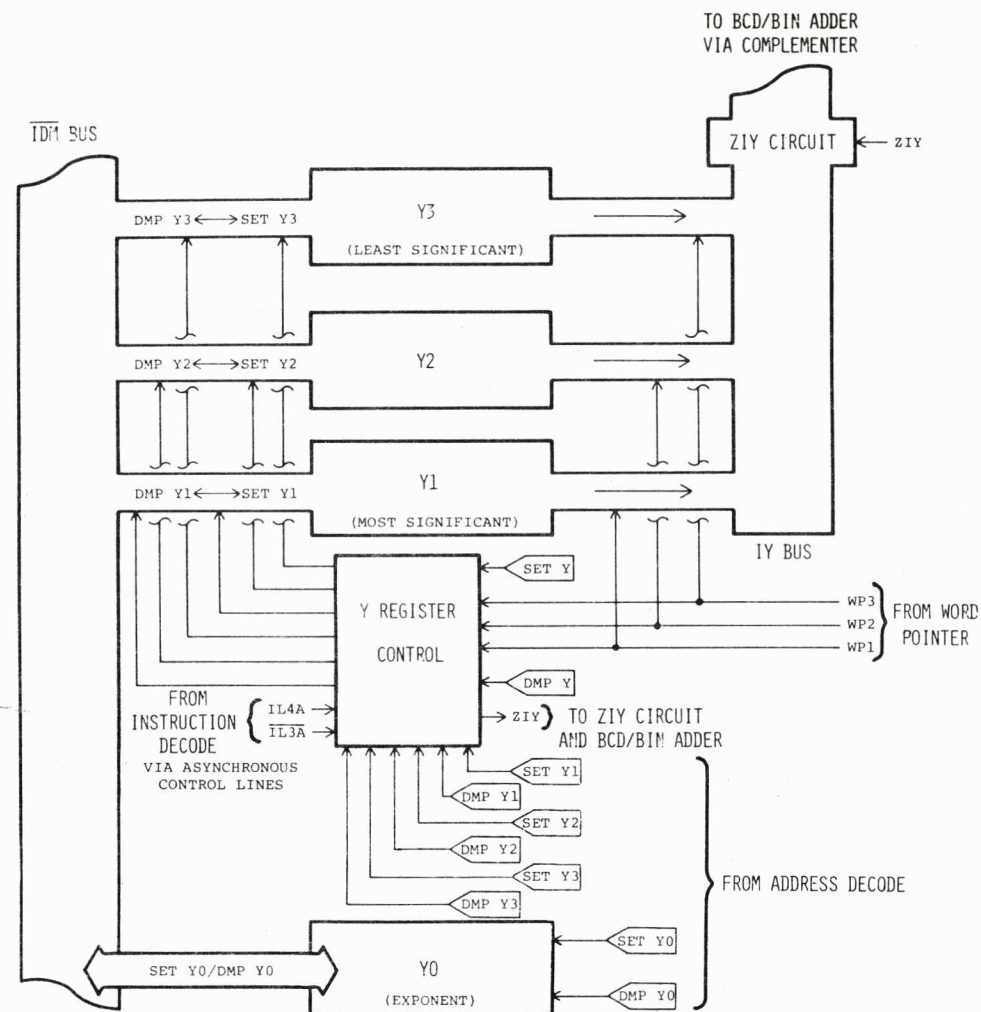


FIG 8-1

SECTION 3

Figure 8-1 is an overview of the Y registers and of Y Register Control. The four Y registers comprise AR2. Y0 is the exponent word and Y1-Y3 contain the mantissa. AR2 is used as an operand in arithmetic operations involving AR1. As in the case of the X registers, the mantissa portions of the Y register are made available to the Adder. This is done by having the Word Pointer select one of Y1-Y3 to be placed on the IY Bus. This Bus proceeds through a ZIY circuit to the Adder. The ZIY circuit can force the IY Bus to zeros. Observe that Y0 is not a part of the

selection mechanism implemented with the Word Pointer.

The purpose of Y Register Control is to convert the general purpose SET Y and DMP Y into particular set and dump instructions referencing the register selected by the value of the Word Pointer. Observe that Y0 cannot be selected by this process. Also observe that the only explicit set and dump micro-instructions originate with Address Decode. These sets and dumps are not used as part of the algorithmic processes for doing arithmetic. They are used strictly in response to memory cycles.

DETAILS OF Y REGISTER CONTROL

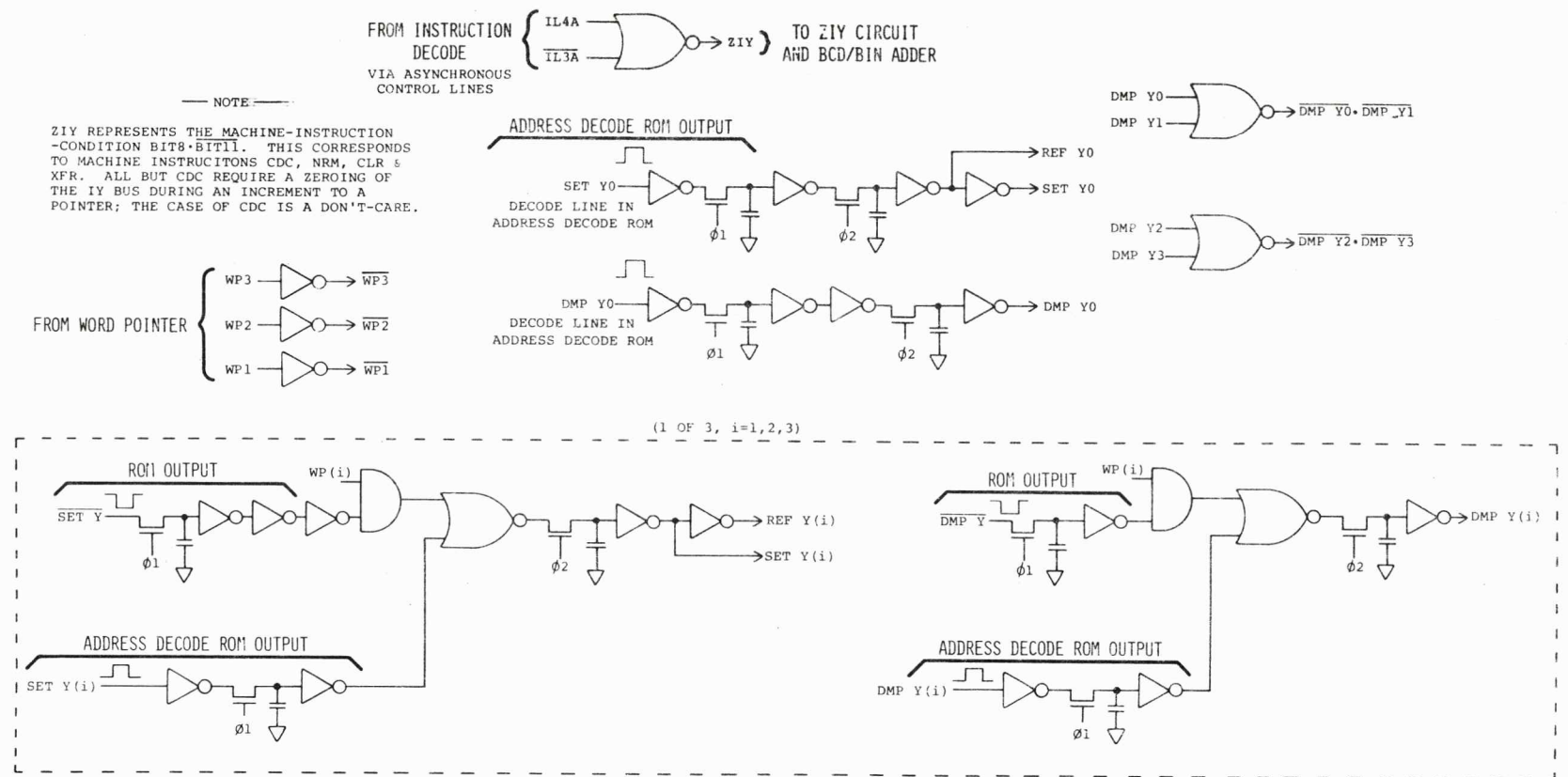


FIG 8-2

Figure 8-2 shows the details of Y Register Control. The drawing is pretty much self-explanatory.

Figure 8-3 shows the details of the actual Y register and the ZIY circuit. It ain't hard to see how this stuff works.

Y REGISTER CONTROL

Y REGISTER OVERVIEW

DMP ADRI & WORD POINTER ENCODER

WORD POINTER SHIFT REGISTER

DETAILS OF THE Y REGISTERS AND THE ZIY CIRCUIT

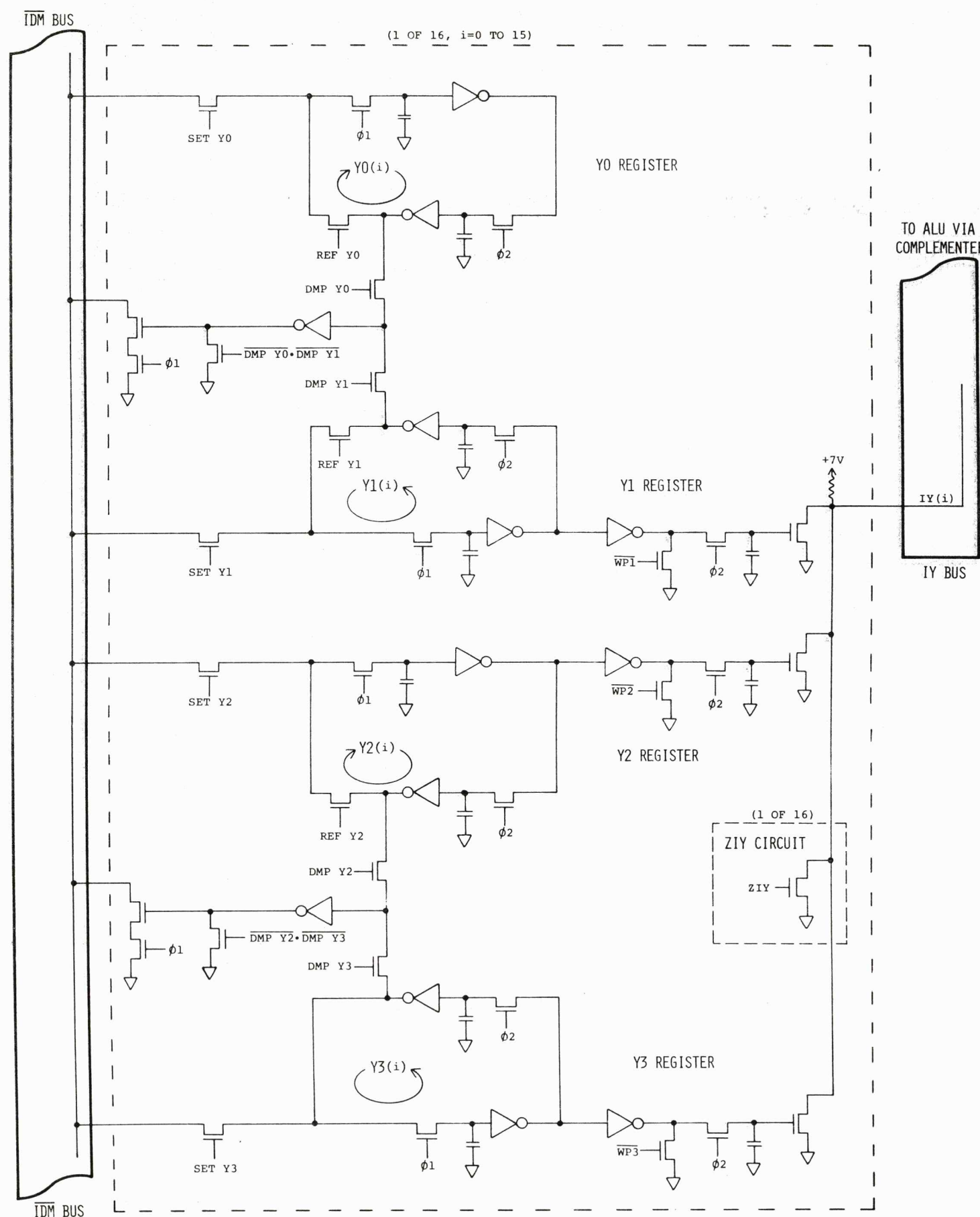


FIG 8-3

SECTION 9

Figure 9-1 is an overview of the BCD/Binary Adder and Complementer, including the DC register. The primary function of the Adder is to perform the BCD summations required for the BCD arithmetic machine-instructions. The ability to perform binary summations is included as part of the means to perform binary multiplication using Booth's algorithm. One might argue that the entire Adder mechanism, including the Complementer, is basically a binary mechanism with a bunch of extra stuff hung on it to make it work in BCD. Well, maybe so, but the usual mode of operation is in BCD and the general circumstances surrounding the use of the thing in binary are not all that obvious.

The Adder adds the contents of the IX Bus to the contents of the IY Bus. No special signals are required to cause the Adder to function; it is constantly responding to its inputs and the current state of its control signals. The output of the Adder is obtained through the micro-instruction DMP ADD. This places the resulting sum on the IDB Bus.

As mentioned above, the inputs to the Adder are the IX Bus and IY Bus. The IX Bus will represent either of X1, X2 or X3. The IY Bus similarly represents either of Y1, Y2 or Y3. The selection as to which is made by the Word Pointer, and the X's and Y's will always be in agreement as to their numerical designator. The Y inputs are fed through the Complementer. This can provide the BCD complement, the binary complement or the uncomplemented value, of the IY Bus, to the Adder. This is accomplished with the control signals BCD COMP, BIN COMP and TRUE, respectively. The Adder itself uses the signal BCD to determine whether to add in binary or BCD. This signal converts the binary summation process to one of BCD, in a manner described in some detail below. The other input to the Adder is the signal CIN₀. This is the initial carry-in. During BCD operations this carry-in is controlled by the DC register.

OVERVIEW OF THE BCD/BIN ADDER AND COMPLEMENTER

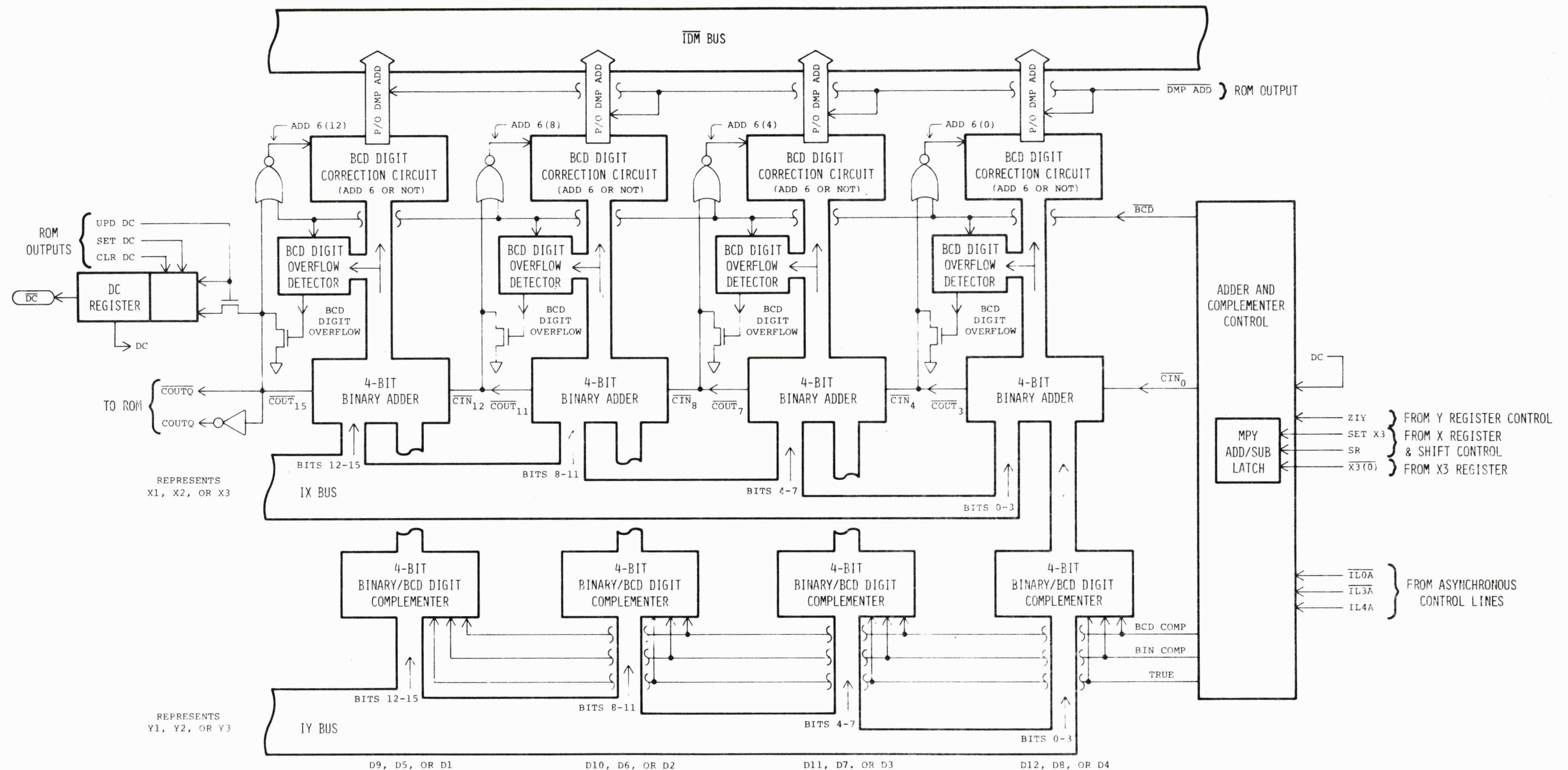


FIG 9-1

SECTION 9 (CONTINUED)

During the multiply with Booth's algorithm it is more complicated.

The various control signals to the Adder are managed by the circuitry in Adder And Complementer Control. The properties of these control signals are explained in the notes to Figure 9-2-1, and need not be repeated here.

The carry-out from the Adder is available both as a qualifier to the ROM (COUTQ and COUTQ) and as an input to control the setting of the DC register. The micro-instruction

UPD DC causes the DC register to assume the value of the carry-out. DC can also be explicitly set and explicitly cleared with the micro-instructions SET DC and CLR DC, respectively. The DC register is used to drive the output pad DC, as well as to supply the potential carry-in input to the next segment of a BCD summation.

As the general structure of the drawing reveals, the Adder is partitioned into four 4-bit mechanisms. This is an obvious accom-

modation to BCD arithmetic. There is nothing particularly tricky about the way in which the Complementer works, and its detailed drawings should make its operation abundantly clear. The 4-bit Binary Adder is just that. The means used are nearly identical to those employed in the BPC. The 4-bit Binary Adder is made into a BCD Adder by the addition of a BCD Digit Overflow Detector and a BCD Digit Correction Circuit. The 4-bit Binary Adder will add BCD numbers as if they were two binary

numbers. These latter two circuits will correct the errors produced when the binary addition fails to produce results that are legitimate for BCD. This involves forcing a carry-out and adding six to the binary sum. For binary addition the BCD Digit Overflow Detectors are disabled, and the BCD Digit Correction Circuit is forced to always add zero. All of this is thoroughly explained in the notes for the detailed drawing of the BCD Digit Correction Circuit.

BCD/BIN ADDER,
COMPLEMENTER & DC
OVERVIEW

Y REGISTERS
& ZIY CIRCUIT

DETAILS OF ADDER & COMPLEMENTER CONTROL, AND MPY ADD/SUB LATCH

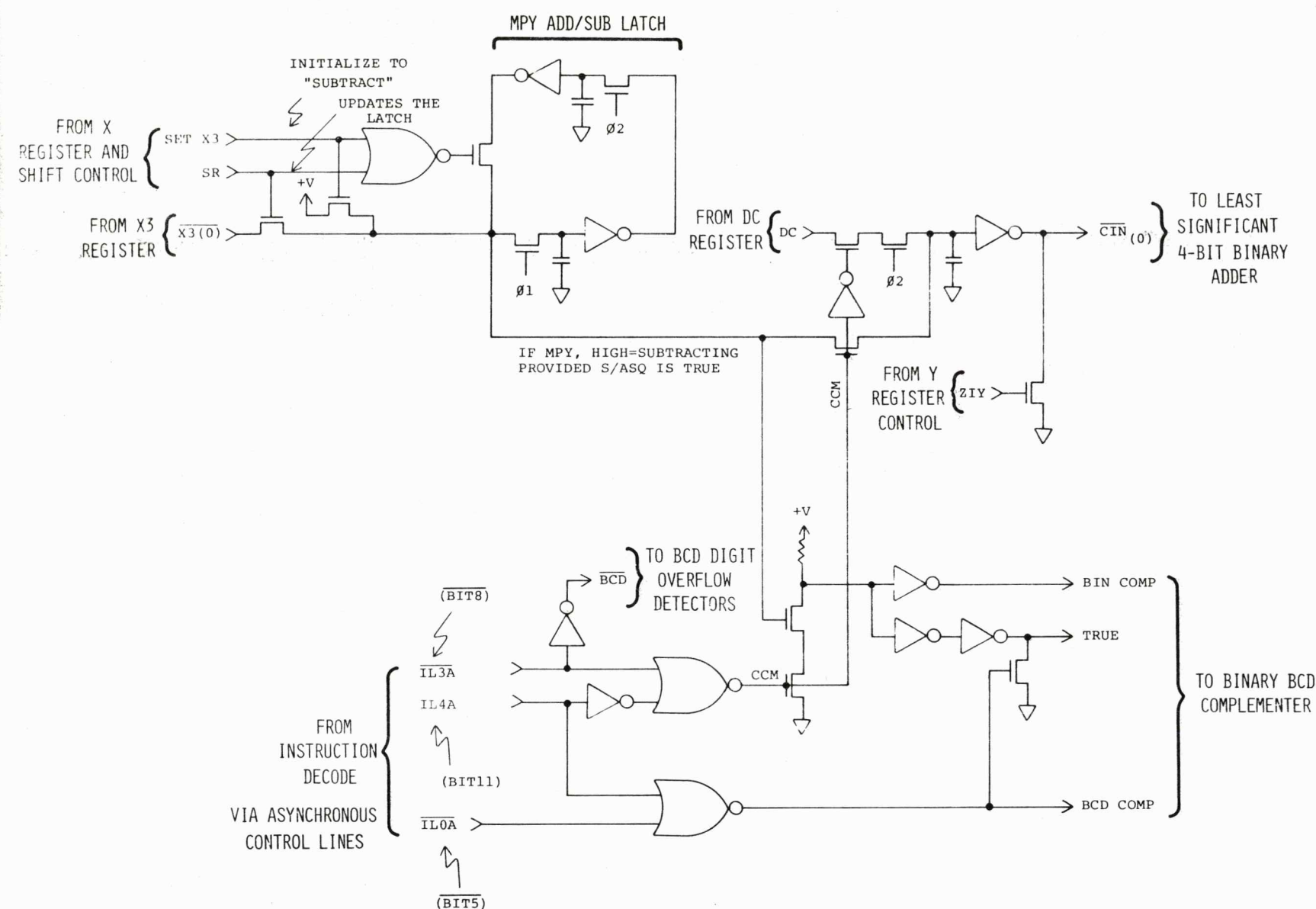


FIG 9-2

NOTES

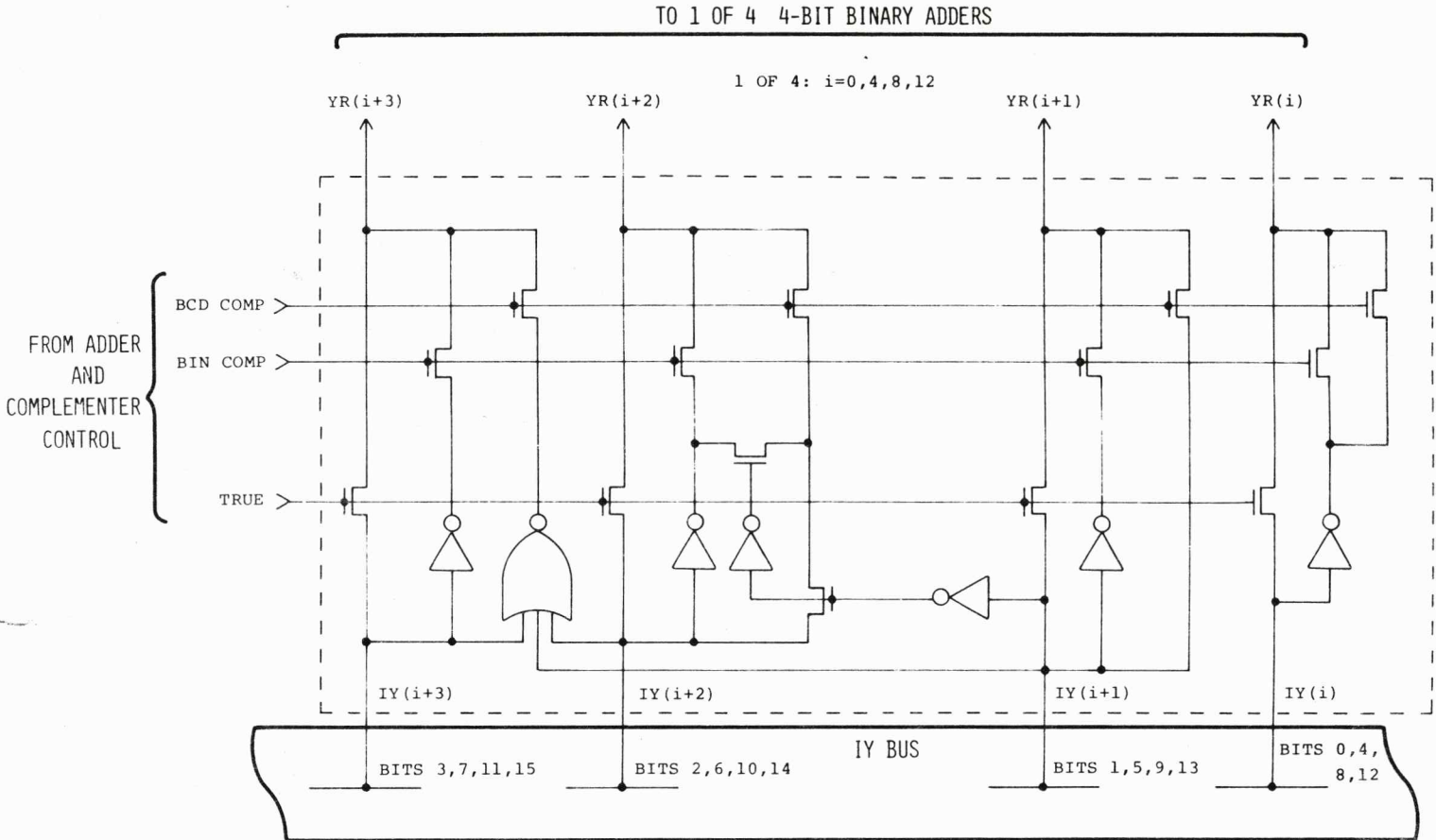
1. IL3A REPRESENTS MACHINE-INSTRUCTION BIT8. BIT8 IS A ZERO ONLY FOR THE BCD ARITHMETIC MACHINE-INSTRUCTIONS: FXIA, MWA, CMX, CMY, FMP AND FDV. \overline{BCD} IS HIGH ONLY DURING THOSE INSTRUCTIONS.
2. IL0A REPRESENTS MACHINE-INSTRUCTION BIT5 AND IL4A REPRESENTS MACHINE-INSTRUCTION BIT11. ONLY MACHINE-INSTRUCTIONS CMX & CMY HAVE THE PATTERN BIT11·BIT5. BCD COMP REPRESENTS THOSE INSTRUCTIONS ONLY.
3. Ziy REPRESENTS MACHINE-INSTRUCTION CONDITION BIT8· $\overline{BIT11}$. THIS PATTERN OCCURS IN CDC, CLR, XFR & NRM. ALL BUT CDC INVOLVE INCREMENTS BY ONE TO A COUNTER IN ONE OF THE X REGISTERS. THE ONE IS GENERATED BY ZEROING THE IY BUS AND GENERATING A CARRY IN TO THE ADDER. IN FACT, THE ONLY TIME IT IS ZEROED IS WHEN GENERATING SUCH AN INCREMENT. THE CASE OF CDC IS A DON'T-CARE.
4. CCM REPRESENTS MACHINE-INSTRUCTION CONDITION BIT8·BIT11. THIS CORRESPONDS TO THESE MACHINE-INSTRUCTIONS: MPY, MRX, DRS, MLY & MRY. OF THOSE, ALL BUT MPY ARE DON'T-CARES. CCM IS PART OF A CONDITIONAL BIN COMP BASED ON THE ADD/SUB LATCH.
5. BIN COMP IS THE AND OF CCM AND A HIGH ADD/SUB LATCH. THIS REPRESENTS THE "PARTIAL PRODUCT SUBTRACTION" OPERATION FOUND IN BOOTH'S ALGORITHM (USED BY MPY). THAT IS THE ONLY TIME BIN COMP IS EVER USED.
6. TRUE IS NORMALLY THE OPPOSITE OF BIN COMP (WHICH IS "NORMALLY" FALSE), EXCEPT WHEN BCD COMP IS HIGH. THEN TRUE IS FORCED FALSE. BIN COMP (REQUIRES CCM WHICH REQUIRES MACHINE-INSTRUCTION BIT11) AND BCD COMP (REQUIRES MACHINE-INSTRUCTION BIT11) ARE MUTUALLY EXCLUSIVE EVENTS.
7. ORDINARILY, DC CONTROLS CIN(0) UNLESS CIN(0) IS FORCED BY Ziy. IF CCM IS HIGH, HOWEVER, THE ADD/SUB LATCH CONTROLS CIN(0). IF CCM IS HIGH, Ziy WILL NOT FORCE CIN(0), AS CCM & Ziy ARE MUTUALLY EXCLUSIVE AROUND MACHINE-INSTRUCTION BIT11.

SECTION 9 (CONTINUED)

Figure 9-2 shows the details of Adder And Complementer Control and of the MPY ADD/SUB Latch. The notes on the drawings provide an explanation of this circuitry.

Figure 9-3 shows the details of the Binary/BCD Complementer. The table indicates the relationship established by the Complementer. It's relatively clear how the circuitry implements the definition established by the table.

DETAILS OF THE BINARY/BCD COMPLEMENTER



EACH BIT ALWAYS OPPOSITE

DIGIT	BCD 9'S COMPLEMENT	TRUE BINARY	BINARY 1'S COMPLEMENT	#
0	1 0 0 1	0 0 0 0	1 1 1 1	0
1	1 0 0 0	0 0 0 1	1 1 1 0	1
2	0 1 1 1	0 0 1 0	1 1 0 1	2
3	0 1 1 0	0 0 1 1	1 1 0 0	3
4	0 1 0 1	0 1 0 0	1 0 1 1	4
5	0 1 0 0	0 1 0 1	1 0 1 0	5
6	0 0 1 1	0 1 1 0	1 0 0 1	6
7	0 0 1 0	0 1 1 1	1 0 0 0	7
8	0 0 0 1	1 0 0 0	0 1 1 1	8
9	0 0 0 0	1 0 0 1	0 1 1 0	9
10	0 1 0 1	0 1 0 1	1 0 1 0	10
11	0 1 0 0	0 1 0 0	1 0 1 1	11
12	0 0 1 1	0 0 1 1	1 1 0 1	12
13	0 0 1 0	0 0 1 0	1 1 0 0	13
14	0 0 0 1	0 0 0 1	1 1 1 1	14
15	0 0 0 0	0 0 0 0	1 1 1 0	15

TRUE BIT1
TRUE BIT2
TRUE BIT3

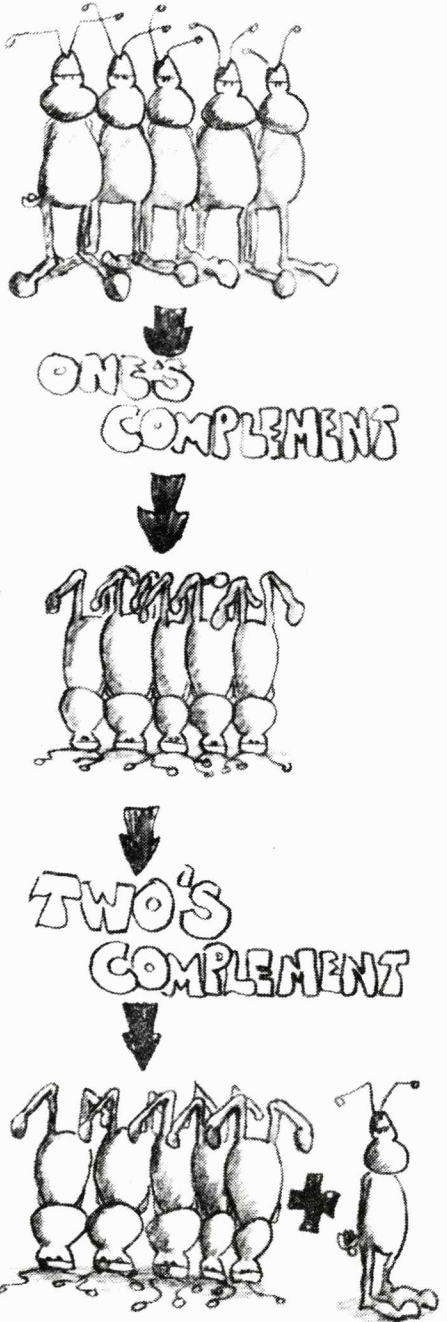
COMP BIT3

ALWAYS THE SAME

ALWAYS OPPOSITE

COMP BIT2 = TRUE BIT2
IF BIT1 = 0
COMP BIT2 = TRUE BIT2
IF BIT1 = 1

FIG 9-3

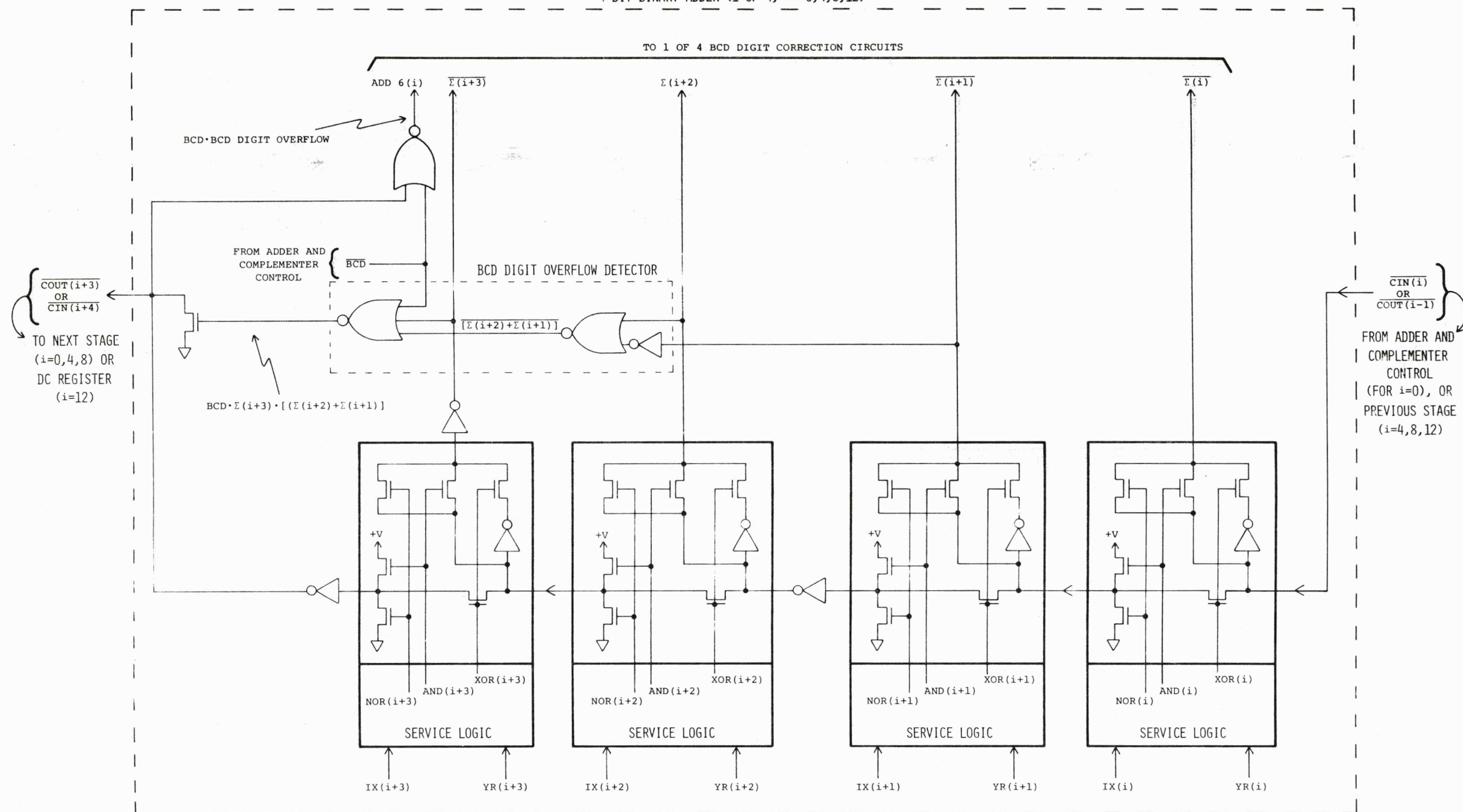


BINARY/BCD
COMPLEMENTER

ADDER &
COMPLEMENTER CONTROL,
MPY ADD/SUB LATCH

DETAILS OF THE 4-BIT BINARY ADDERS AND BCD DIGIT OVERFLOW DETECTORS

4-BIT BINARY ADDER (1 OF 4, $i = 0, 4, 8, 12$)



IX's ARE FROM THE IX BUS
YR's ARE THE OUTPUTS OF THE BINARY/BCD COMPLEMENTER

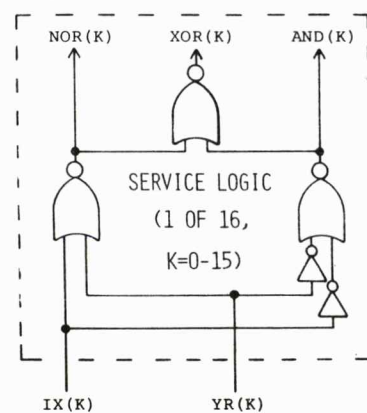
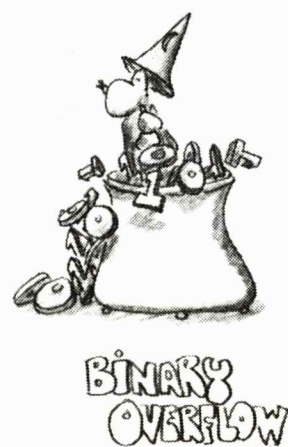
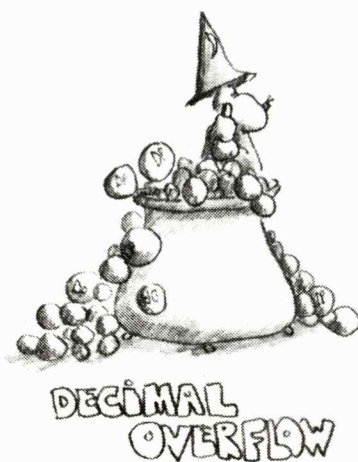


FIG 9-4-1

SECTION 9 (CONTINUED)

Figure 9-4-1 shows the details of the 4-bit Binary Adder and of the BCD Digit Overflow Detectors. The Adder sections are quite similar to those employed in the BPC. The process of overflow detection is described in the notes of Figure 9-4-3.

DETAILS OF THE BCD DIGIT CORRECTION & DMP ADD CIRCUITS

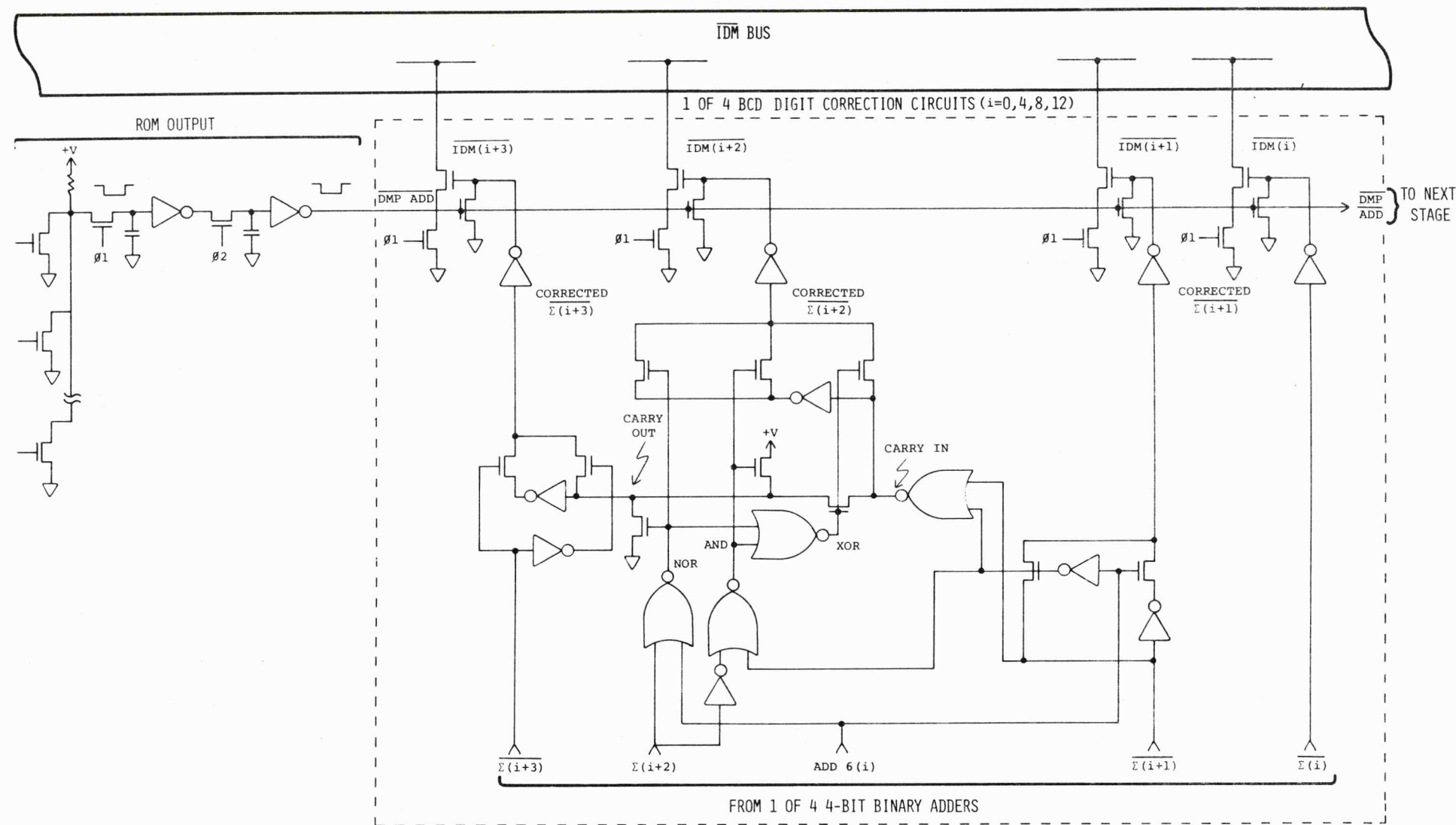


FIG 9-4-2

SECTION 9 (CONTINUED)

Figure 9-4-2 shows the details of the BCD Digit Correction and DMP ADD circuits. Digit correction involves adding six or zero to a BCD digit. As such, this is a special adder and its operation is explained in the notes of Figure 9-4-3. There is nothing remarkable about the DMP ADD circuit.

Figure 9-4-3 is the notes for the Figures 9-4-1 and 9-4-2.

NOTES ON BCD ADDITION

- AS LONG AS THE SUM OF TWO BCD DIGITS IS NINE OR LESS, THEIR BCD SUM EQUALS THEIR BINARY SUM. HOWEVER, THIS MAXIMUM ALLOWABLE BCD SUM OF NINE IS SIX LESS THAN THE MAXIMUM POSSIBLE SUM PRODUCABLE BY BINARY ADDITION. TO COMPENSATE FOR THIS DIFFERENCE, BCD ADDITIONS THAT PRODUCE OUT-OF-RANGE BINARY RESULTS ARE ADJUSTED BY ADDING SIX TO EACH SUCH DIGIT. THIS PRODUCES PROPER CARRIES AND PROPER DIGIT ROLL-OVER.

- A BINARY SUM GREATER THAN NINE IS DETECTED AS SHOWN BELOW. THIS BIT IS SET $\overline{1} \ 0 \ 0 \ 1$ AND EITHER OF THESE BITS IS SET THAT IS:

$$\text{BCD} \cdot \overline{E(i+3)} \cdot [E(i+2) + E(i+1)] = 1$$

NOW, BCD DIGIT OVERFLOW ALSO NEEDS TO BE ACCOMPANIED BY A CARRY OUT, WHICH IN BINARY ADDITIONS IS GENERATED BY NATURAL PROCESS, BUT BCD OVERFLOW NEED NOT ENTAIL BINARY OVERFLOW, SO BCD OVERFLOW MUST FORCE A CARRY OUT.

- NOW, $(\text{CARRY OUT}) \cdot (\text{BCD ADDITION}) = 1$ IMPLIES THE NEED TO ADD 6 TO THAT DIGIT. HENCE:

$$\text{ADD } 6(i) = \text{BCD} \cdot \text{COUT}$$

- TO ADD 6:

$$\begin{array}{r} E(i+3) \quad E(i+2) \quad E(i+1) \quad E(i) \\ + \quad 0 \quad 1 \quad 1 \quad 0 \\ \hline \end{array}$$

BIT PATTERN OF OVERFLOWED BCD DIGIT
—SIX
GENERATED BY ADD 6(i) BEING A ONE

- LEAVE $E(i)$ ALONE.
- $E(i+1)$ GETS COMPLEMENTED IF IT IS A ONE, IN WHICH CASE, ALSO GIVE A CARRY-OUT TO ADDITION OF NEXT BIT OVER, (STEP iii).
- $E(i+2)$, 1, AND THE CARRY FROM STEP ii ARE ADDED BY A FULL ADDER, PRODUCING A PROPER CARRY FOR THE ADDITION OF THE NEXT BIT OVER, (STEP iv).
- $E(i+3)$, 0, AND THE CARRY FROM STEP iii ARE ADDED WITHOUT REGARD TO FURTHER CARRIES. (THE MAIN BINARY ADDITION AND THE OVERFLOW DETECTION PROCESS TOOK CARE OF THE MAIN CARRY OUT FOR THIS DIGIT.) THIS ADDITION AMOUNTS TO COMPLEMENTING $E(i+3)$ IF THE CARRY FROM STEP iii IS A ONE. FOR SOME REASON THE HARDWARE INSTEAD USES THE CARRY FROM STEP iii AS THE OUTPUT, COMPLEMENTING $E(i+3)$. THE METHODS ARE EQUIVALENT, HOWEVER.

- IN THE EVENT THAT ADD 6(i) IS A ZERO, $E(i+1)$, $E(i+2)$ AND $E(i+3)$ ARE NOT SIMPLY ROUTED AROUND THE "ADD 6" MECHANISM, THE ADDITION IS STILL PERFORMED, BUT SHOWN BELOW.

$$\begin{array}{r} E(i+3) \quad E(i+2) \quad E(i+1) \quad E(i) \\ + \quad 0 \quad 0 \quad 0 \quad 0 \\ \hline \end{array}$$

GENERATED BY ADD 6(i) BEING A ZERO

OF COURSE, IN EITHER CASE, $E(i)$ IS ROUTED STRAIGHT THROUGH THE CORRECTION CIRCUIT.

- THUS THE BCD DIGIT CORRECTION CIRCUIT CAN BE CHARACTERIZED AS A SIMPLIFIED FOUR-BIT ADDER, ONE OF WHOSE INPUTS IS THE UNCORRECTED BIT PATTERN, AND WHOSE OTHER INPUT IS 0110 IF ADD 6(i) IS A ONE, AND 0000 IF ADD 6(i) IS A ZERO.

FIG 9-4-3

NOTES ON
BCD DIGIT CORRECTION
& DMP ADD CIRCUITS
BCD ADDITION

4-BIT BINARY ADDERS
& BCD DIGIT
OVERFLOW DETECTORS

DETAILS OF THE DECIMAL CARRY REGISTER

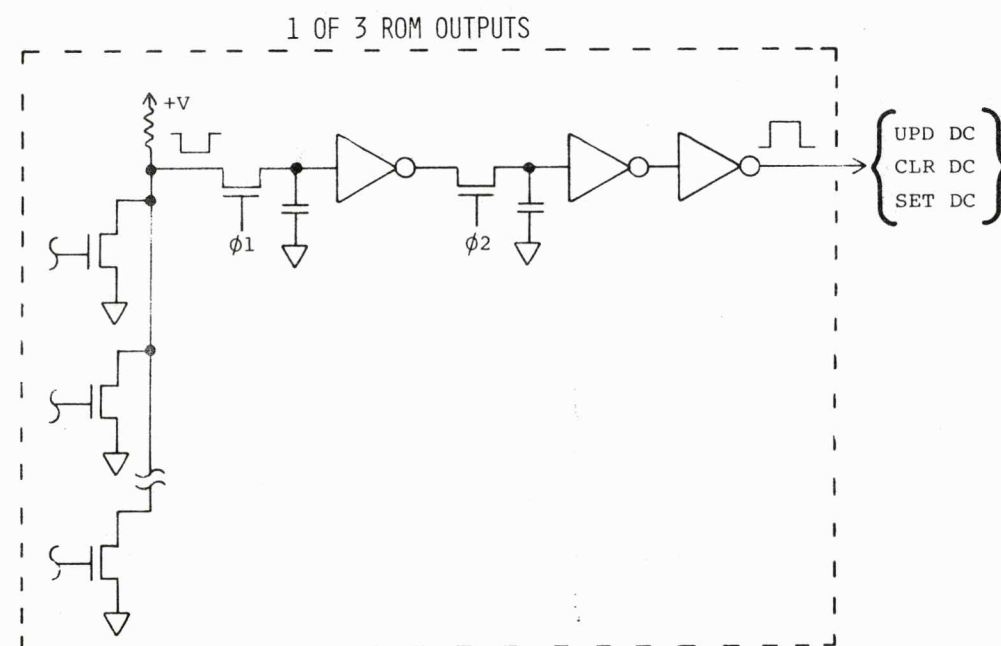
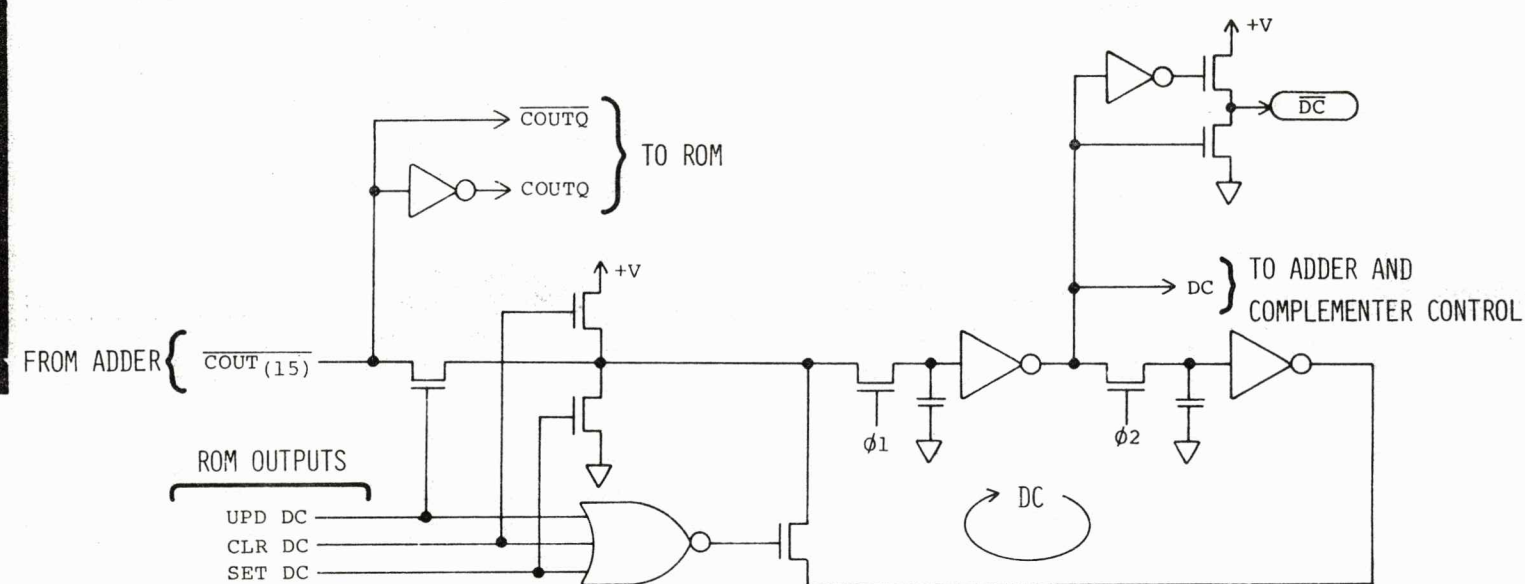
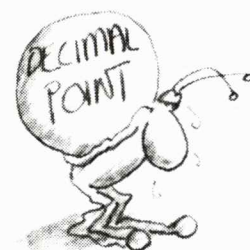


FIG 9-5

SECTION 9 (CONTINUED)

Figure 9-5 shows the details of the Decimal Carry register (DC). Observe how it can be updated by the value of COUT(15), explicitly set, or, explicitly cleared.



DECIMAL CARRY

THE DMP ONE CIRCUIT

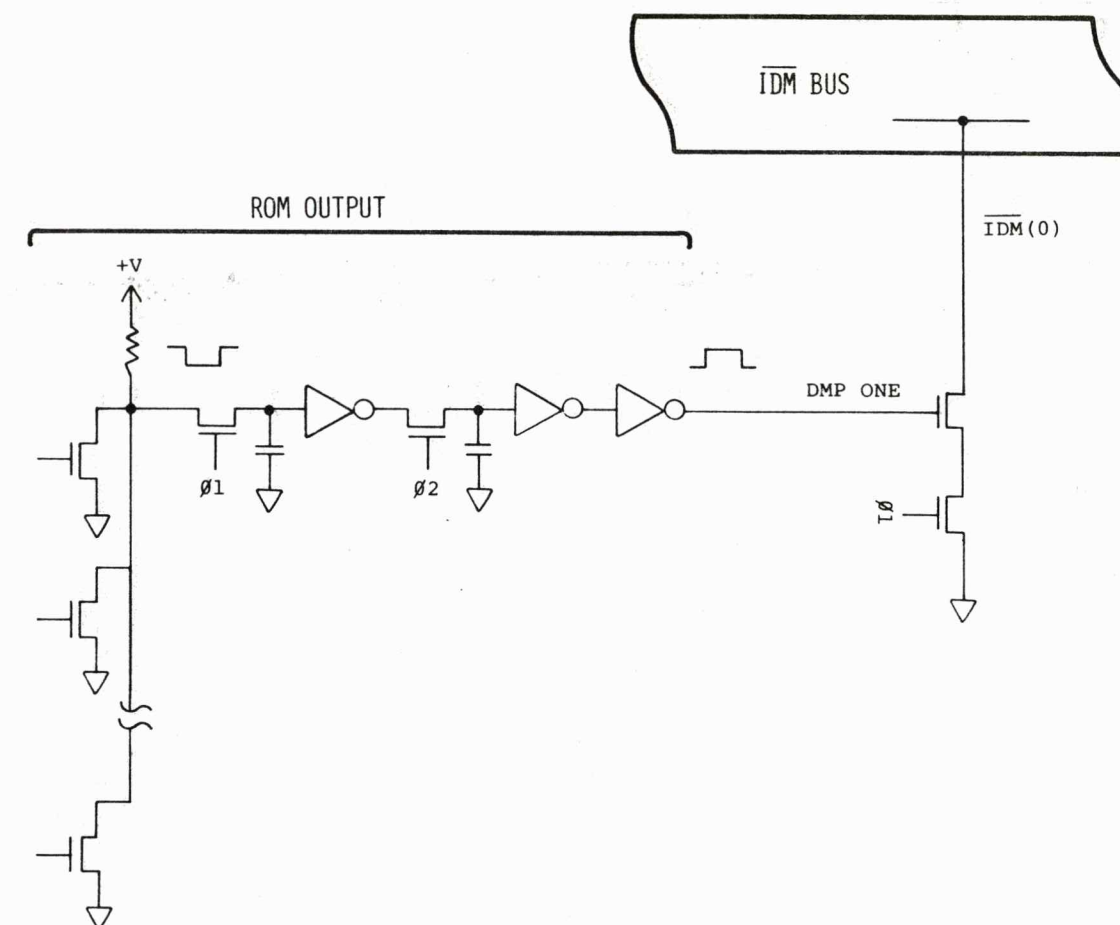
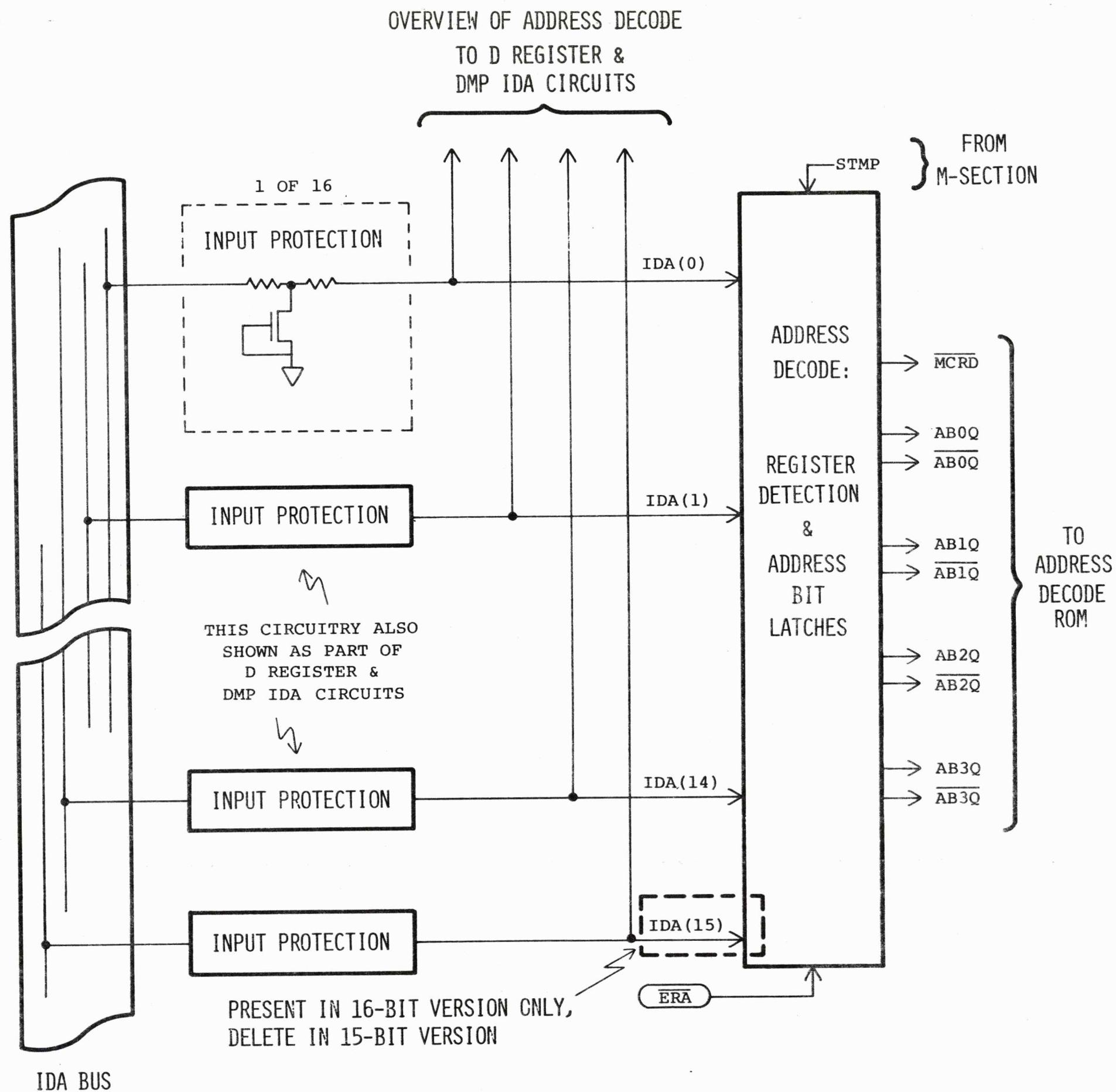


FIG 10-1

SECTION 10

Figure 10-1 shows the details of the DMP ONE circuit. The purpose of this micro-instruction is to generate the address of the B register onto the IDM Bus. This is required because certain machine-instructions require memory cycles directed to the B register. They also require memory cycles directed to the A register. But its address is easy to obtain since it's all zeros. All that is necessary is to SET the D register with no corresponding DMP.



SECTION 11

Figure 11-1 is an overview of Address Decode. It monitors the IDA Bus and the signal representing Start Memory (STMP). It generates signals that represent the occurrence of a memory cycle directed to the EMC. In particular, it generates a signal MCRD which is used in the setting of the Read and Write Register Latches of the M-Section. What MCRD means is that the EMC is the object of a memory cycle. The other signals produced by Address Decode are qualifiers generated from the captured address bits of the address involved in the memory cycle. Address Decode also implements some ERA mode addressing. One of the Address Bit Latches is enabled only during ERA mode operation. ERA addressing for the EMC is described, along with the Address Decode ROM, in the next section.

ADDRESS DECODE
OVERVIEW

DMP ONE
CIRCUIT

DECIMAL CARRY
REGISTER

PART OF ADDRESS DECODE; REGISTER DETECTION CIRCUIT & ADDRESS BIT LATCH CIRCUITS

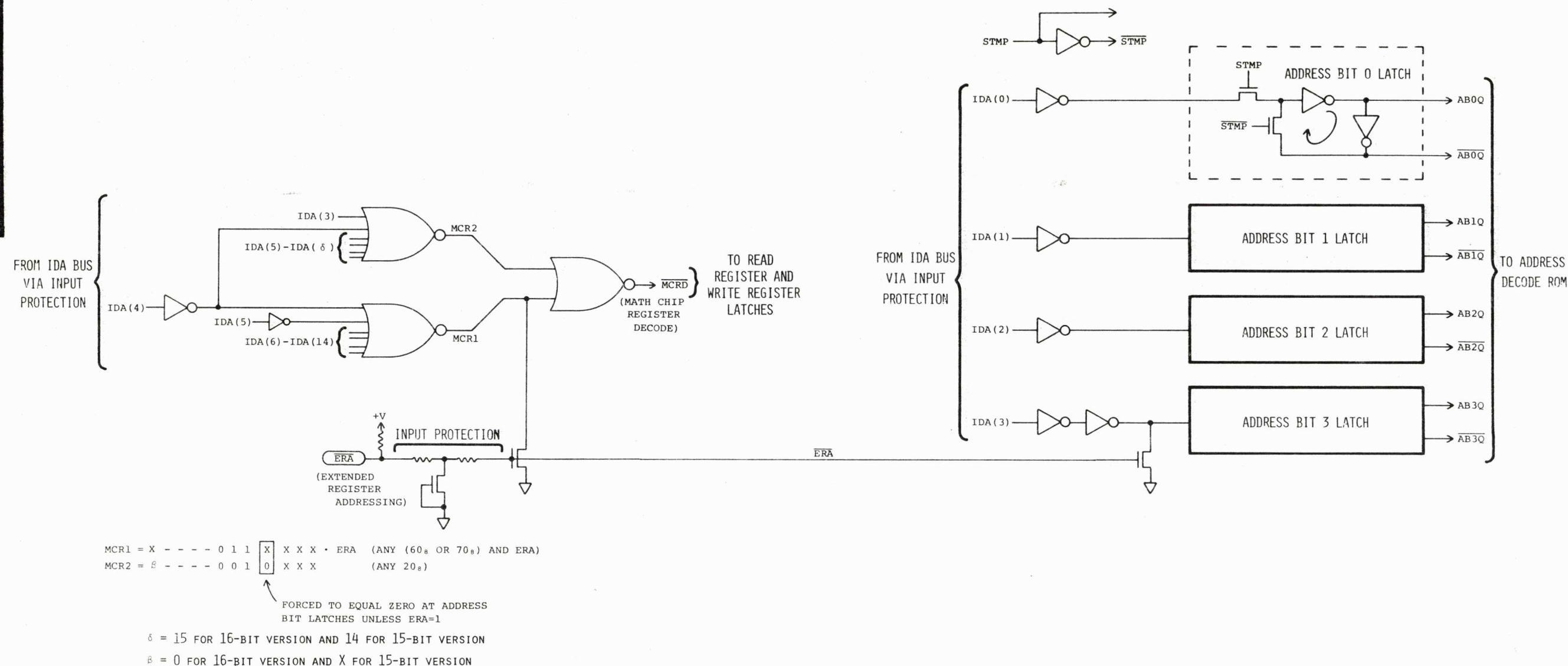


FIG II-2

SECTION 11 (CONTINUED)

Figure 11-2 shows the details of Address Decode. A series of gates are used to detect addressing associated with the EMC. MCR1 represents the collection of ERA mode addresses. MCR2 represents the occurrence of a register address contained by the EMC. These two signals are NOR'd together to produce MCRD. The differences between the 15 and 16-bit versions involve only the additional address bit.

Observe how the Address Bit Latches are enabled by STMP. STMP is

a pulse, generated by the M-Section, whose leading edge is coincident with the grounding of Start Memory. After one state STMP goes false and the Address Bit Latches are closed, latching the address. Observe that the fourth Address Bit Latch is enabled only during ERA mode addressing. Indeed, a 1 in that position stands for ERA mode addressing. This will be more apparent after inspection of the bit patterns responded to by the Address Decode ROM.

OVERVIEW OF THE ADDRESS DECODE ROM

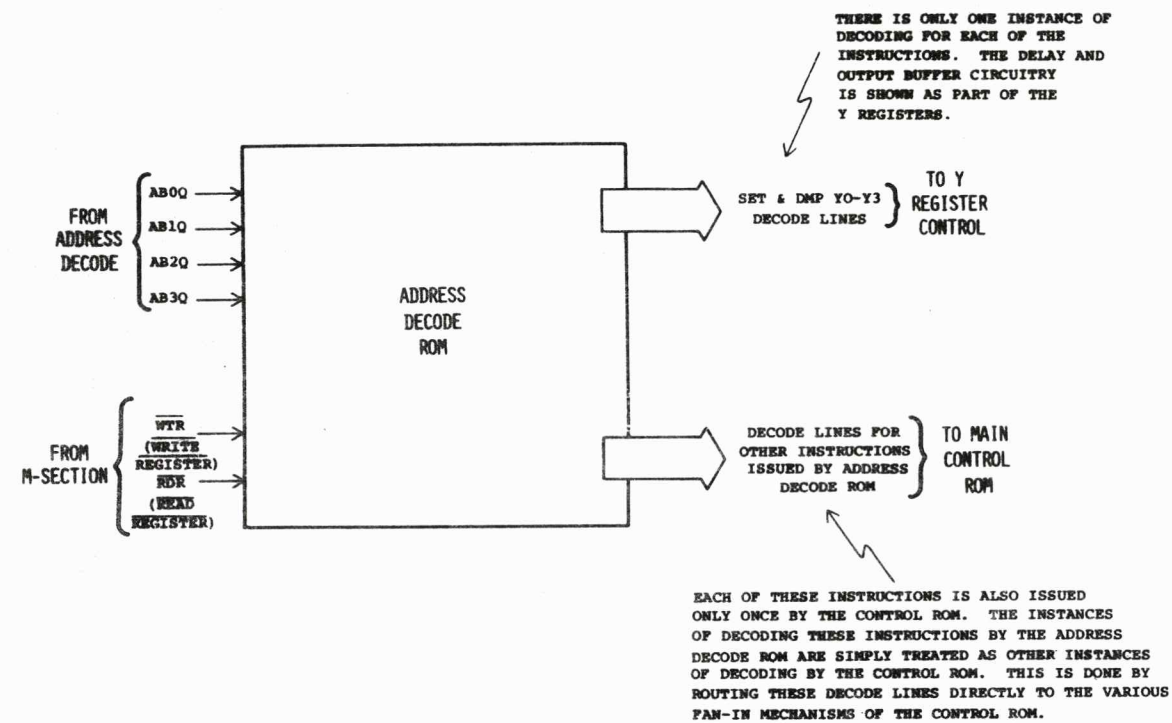


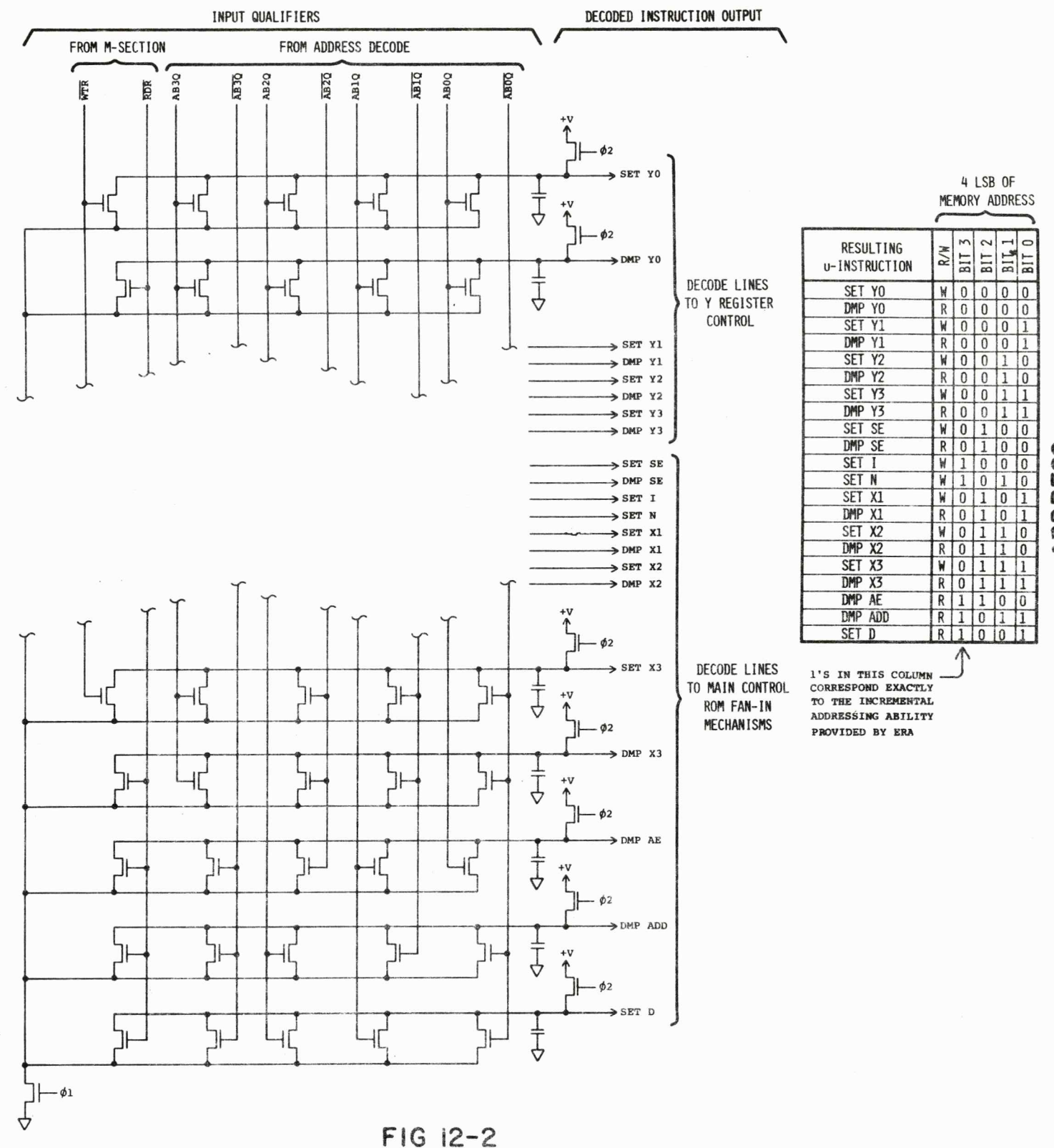
FIG 12-1

SECTION 12

Figure 12-1 is an overview of the Address Decode ROM. The inputs to this ROM are the four address bits and two signals representing the occurrence of memory cycles directed to the EMC. Of these latter two signals, one represents read memory cycles and the other represents write memory cycles. These signals are RDR and WTR. The output of the ROM consists of SET and DMP micro-instructions, most of which are routed to the fan-in mechanism of the main control ROM. The timing and duration of these micro-instructions is controlled by the M-Section. It does this by controlling the onset and duration of WTR and RDR.

Figure 12-2 shows the details of the Address Decode ROM circuitry and lists the various micro-instructions that are decoded for the different input conditions. The ROM mechanism itself is a phase two

DETAILS OF THE ADDRESS DECODE ROM



precharge/phase one discharge type affair. An output line is decoded when all of the transistors between it and the discharge line are turned off. Under these circumstances the phase two precharge lingers into phase

one and causes the output. However, if any of the transistors between the output line and the discharge line are turned on, phase one removes the phase two precharge, resulting in no output.

REGISTER DETECTION
& ADDR. LATCH CIRCUITS

ADDRESS DECODE ROM OVERVIEW

ADDRESS
DECODE ROM

Figure 13-1 is an overview of the M-Section in the EMC. Overall, it is similar to the M-Section in the BPC. The only difference between the 15 and 16-bit versions of the EMC's M-Section concerns the addition of the POP Synchronizer in the 16-bit version.

Consider STM and its associated circuitry. The ROM output for STM incorporates a latch. Accordingly, the ROM needs to issue STM only once for each memory cycle. ODD can prevent the issuance of STM by the EMC. The EMC also listens to the STM line. A one-shot triggered by STM generates STMP (Start Memory Pulse) whose leading edge matches STM. STMP is used to enable the Read and Write Register Latches, one of which will be set if MCRD from Address Decode is true. The setting of those latches (which might not happen if the memory cycle was directed elsewhere than to the EMC) provides qualifiers to the Address Decode ROM which then generates the various sets and dumps needed for response to the memory cycle. The setting of those latches also allows STMP to feed through a delay which ultimately generates SMC. SMC is delayed to generate MEC, which is sent to the ROM and also used to reset various latches.

MCI (Memory Complete Inhibit) is a signal left over from the development of the chip. It was used to induce an indefinite delay in the issuance of SMC. This kept the data on the IDA Bus for as long as desired. Originally, all chips had this mechanism, but it was removed at the onset of production for most. They just never bothered to take it out in the EMC. (Each turn around costs \$\$\$, as well as affording opportunities for mistakes.) MCI is not bonded up.

STMP is AND'ed with the output of the Read Register Latch to create a SET D. If the Read Register Latch is set it means that the originator of the memory cycle is doing a read and that the EMC is the respondent. The EMC must do a DMP of some register, SET D, SET IDA. The dump of the affected register is supplied by the Address Decode ROM. The SET IDA comes from the SET IDA

ROM decode, where it is forced by RDR. The SET D is provided by the AND of STMP and RDR.

STM is also used in the Bus Request circuitry. STM or BGI each prevent a Bus Grant. A successful Bus Grant is detected and made into STP and SSTP. These latter two signals are used in much the same way as their counterparts in the BPC.

Similar to the IOC, the EMC normally keeps Bus Grant grounded (except during its phase one pre-charge) until there is a Bus Request. BGI or an ongoing EMC memory cycle will prevent the release of Bus Grant. Electrically, it is safer to release Bus Grant only when it is necessary to do so, than it is to have to scramble to get it grounded out soon enough for everyone to get the word.

WRITE is a latched ROM output. also. ODD can disable this output, too. The RDW line is monitored for the purposes of setting the Read Register and Write Register Latches.

Observe that SET IDA is disabled by ODD, and also that SET IDA generates PDR. Also notice that RDR forces a SET IDA. This is for the purpose of responding to read memory cycles directed to the EMC, as explained above.

Consider the DMP IDA ROM output. Observe that it is forced by WTR, but disabled by RDR. WTR forces a DMP IDA as part of the M-Section's response to a write memory cycle directed to a register within the EMC. In this case, the EMC must do a DMP IDA in conjunction with setting some register. The SET micro-instruction will come from the Address Decode ROM.

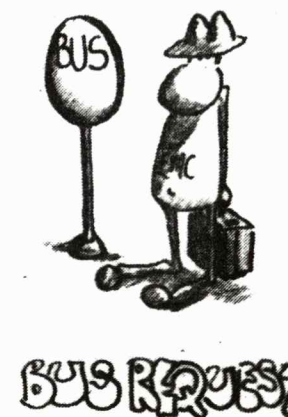
It's probably unnecessary for RDR to disable DMP IDA. The situation to be avoided is as follows. Suppose a Bus Grant occurred while the EMC, in the course of its normal duties, was doing a DMP IDA. Then, further suppose that a read memory cycle was done while BG was true. This would entail a simultaneous SET D, SET IDA and DMP IDA. That would create an unstable situation which

could be resolved by inhibiting the DMP IDA. However, the most well informed opinions hold that no such situation will arise, because a Bus Grant will never be allowed while a memory cycle is in progress. Further, the claim is that the EMC never does a DMP IDA except as part of a memory cycle. The ASM chart does not even contain any don't care DMP IDA's. (That is, DMP IDA's that are decoded superfluously as a result of minimization of the ASM chart.) Therefore, occurrences of DMP IDA and of Bus Requests involving SET IDA are disjoint. Anyhow, it was probably safer to put it in, so they did. Accordingly RDR disables DMP IDA.

SYNC also has a latched ROM output. The latch is reset by Memory Complete, provided that a Bus Grant is not in progress. The idea here is that after SYNC has been given, the next Memory Complete corresponds to the conclusion of the instruction fetch. That's true, provided that there has been no intervening Bus Grant. Had there been a Bus Grant there could be many memory cycles that are not part of the instruction fetch. SYNC would remain true all during such memory cycles.

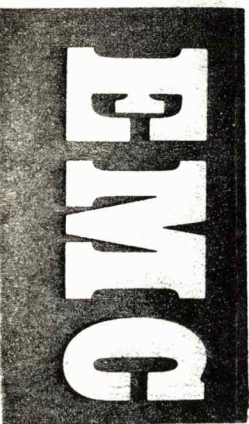
The remaining circuitry concerns POP and the POP Synchronizer. This circuitry is not any different than previous instances of this stuff on the other chips.

Figure 13-2 illustrates the details of the M-Section. It ain't really so bad once you get used to it.



M-SECTION, SET D, SET IDA, DMP IDA, ODD & SYNC CIRCUITS OVERVIEW





DETAILS OF THE M-SECTION, SET D, SET IDA, DMP IDA, ODD AND SYNC CIRCUITS

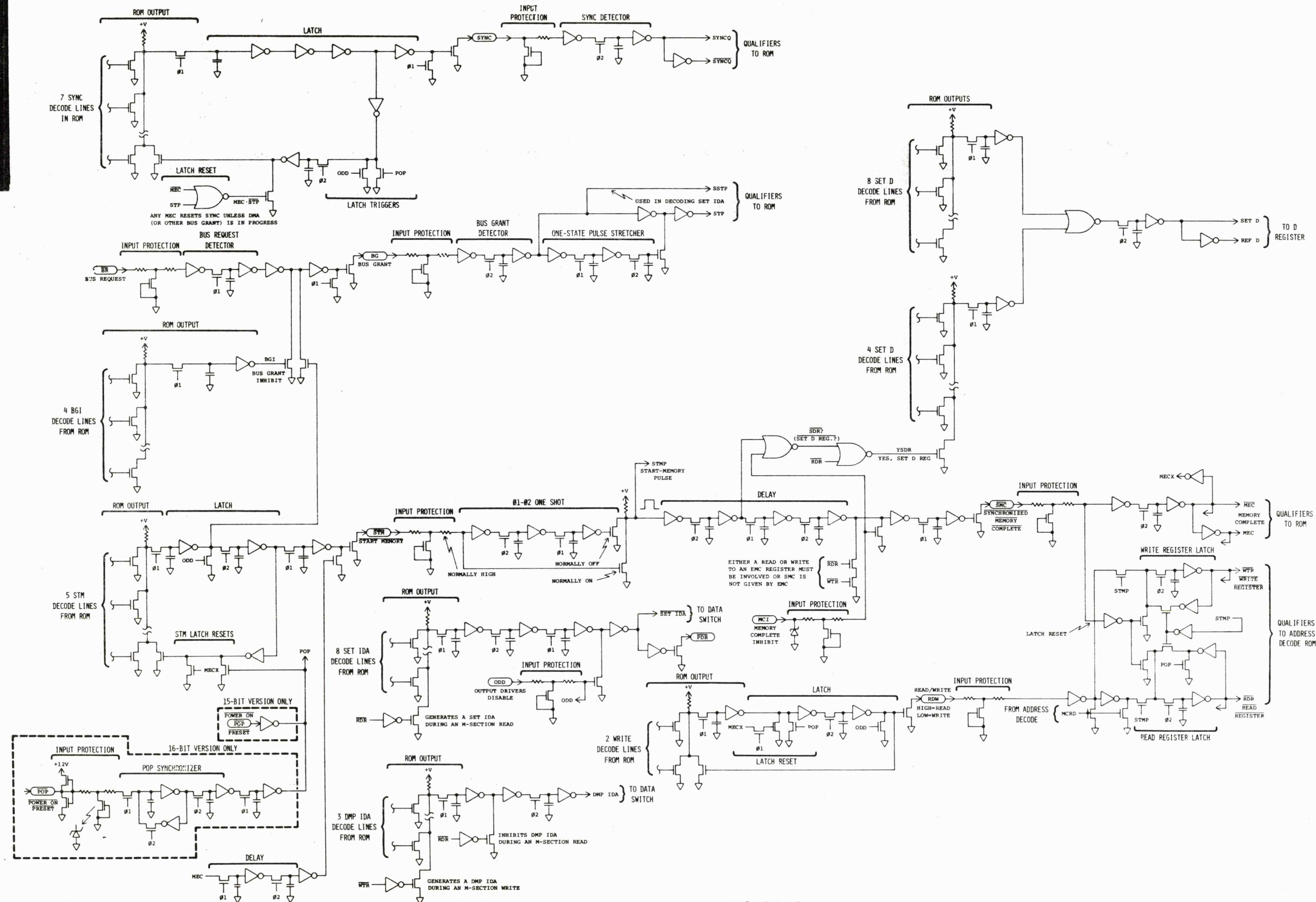


FIG 13-2

INTERPRETING THE EMC ASM CHART

1. What we usually refer to simply as a state ("state 2 for FXA") is generally a coincidence of that particular state-count and some group qualifier encoding pattern (representing a particular group). The most precise way to refer to a location on the ASM chart is indicate both the group and state-counts, (say, B4). Some states (0 and 1) are completely group independent. States are indicated by circles with numbers in them: ④. Group information is prominently displayed next to sections to which it pertains.

2. Each state represents a 02 pre-charge and 01 decode in the ROM. The ASM chart represents what is decoded from the ROM in the various states; it does not necessarily represent end-results that occur simultaneously. If, for instance, two instructions decoded in the same state have different delays between the ROM and the using agency, then they do not result in simultaneous activity, even though they are drawn as being in the same state.

3. Rectangular boxes (SET N) denote micro-instructions. Diamonds (MEC=1?) denote qualifiers affecting the decoding of micro-

instructions. Ovals (STM) denote micro-instructions that are actually decoded and given, but that are "don't-cares". That is, they are present but do not affect the algorithmic process. These don't-cares are the result of minimizing the number of qualifiers used in the ROM.

It is frequently the case that a don't-care instruction is conditional upon one or more qualifiers. For instance, the following notation means that if both D1ZQ and AERQ are true, then INC AE is decoded:

D1ZQ·AERQ:INC AE

It is also frequently the case that several don't-care items occur (unconditionally) in a state. In such an instance the various don't-care micro-instructions are separated by semicolons, thus:

DMP ADD;DMP ADRI

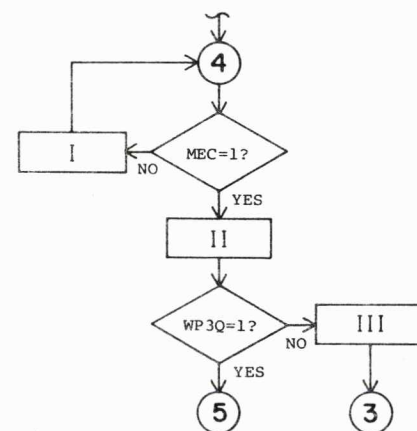
A single instance of using a qualifier or combination of qualifiers can result in but a single instruction. But sometimes a state incorporates two separate uses of the same qualifier in order to decode two instructions. When these instructions are don't-cares, we employ a short-hand notation for this wherein the qualifier is shown once and the instructions are separated by commas, rather than semicolons, thus:

MEC:SET X,DMP ADD;----

rather than:

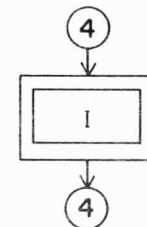
-----;MEC:SET X;
MEC:DMP ADD;-----

4. All activity within a state is decoded and initiated (its delay is begun) at the same time. The fact that a state is shown as a sequential arrangement of boxes and diamonds does not imply sequential activity; the entire state is decoded simultaneously. For example, state 4 of group B is represented below:

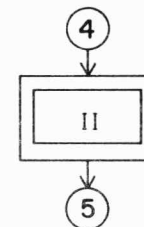


Another way to represent the same activity is illustrated below. We don't draw the ASM chart that way because of the increased size and because of problems in achieving connectedness. Also, overall algorithmic process would be hard to see; the more compact notation results in a more effective visual outline. Within a state however, the expanded notation is often less confusing as it more closely represents the actual way things are done.

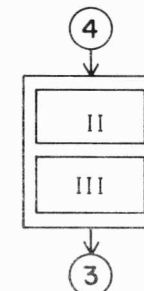
If MEC=0, then:



If MEC=1, and WP3Q=1, then:



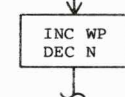
If MEC=1, and WP3Q=0, then:



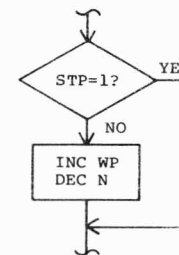
5. Within a state, related instructions are grouped together in the same box solely for the sake of algorithmic clarity.

6. Because of limitations on transistor device size, and the large capacitances of the IDA lines, two consecutive SET IDA's are required to ensure that IDA lines assume their proper final values. If what is being transmitted with the SET IDA is an address for Memory, the STM will accompany the second SET IDA.

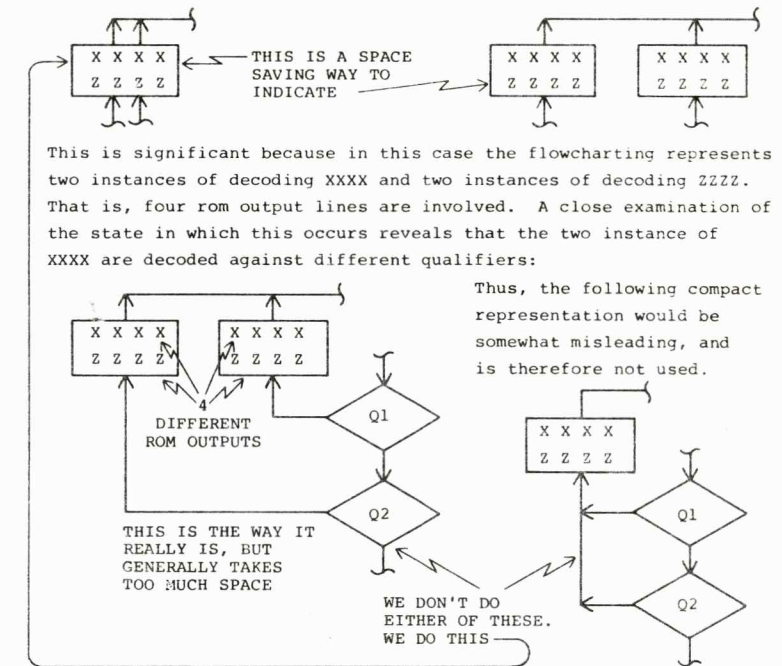
7. Most micro-instructions are accompanied by a vertical bar on either side of their enclosing rectangle:



This is a short hand notation for denoting that (all) the micro-instructions in that rectangle are conditional upon STP (or SSTP), thus:



8. Occasionally the following non-standard flowcharting notation is employed:



ASM CHART
INTERPRETATION

SECTION 14

Figure 14-1 explains how to interpret the EMC's ASM chart. It also explains certain special notational conventions employed in the flow charting depicted in this section.

M-SECTION, SET D, SET IDA,
DMP IDA, ODD & SYNC CIRCUITS

FIG 14-1



OVERVIEW OF THE EMC ASM CHART

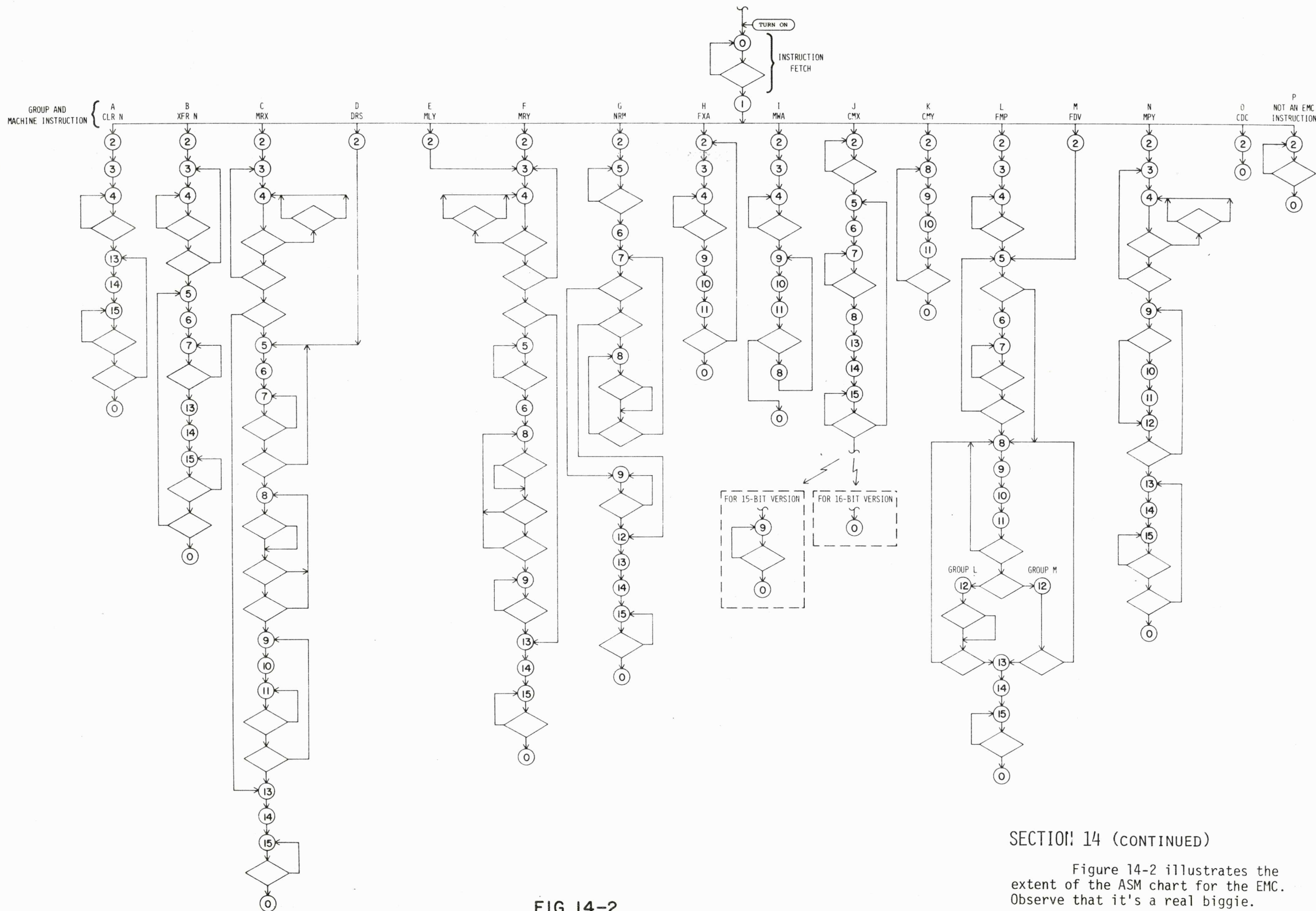


FIG 14-2

SECTION 14 (CONTINUED)

Figure 14-2 illustrates the extent of the ASM chart for the EMC. Observe that it's a real biggie.

INSTRUCTION FETCH AND FAN OUT

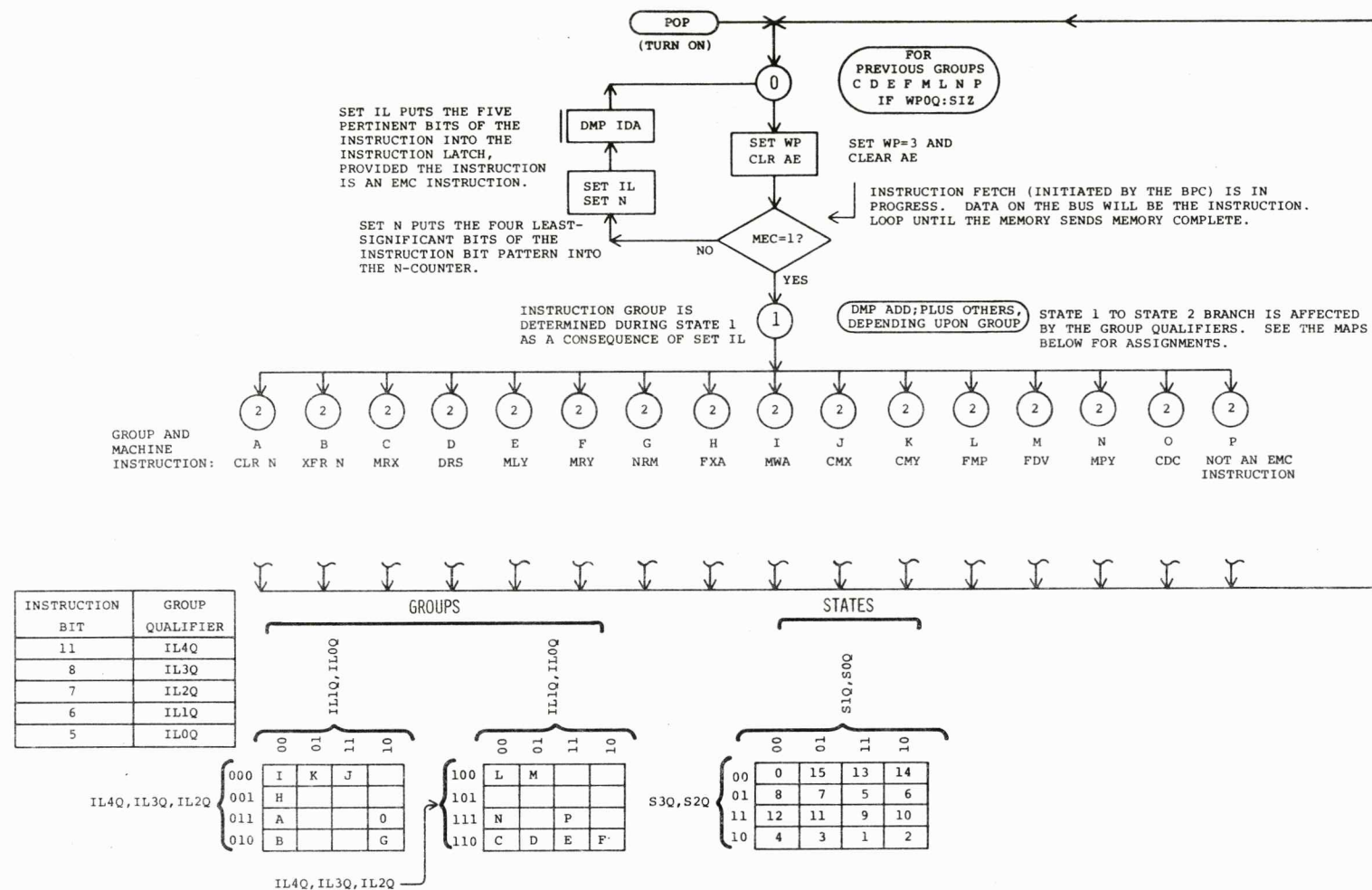


FIG 14-3

SECTION 14 (CONTINUED)

Figure 14-3 illustrates the instruction fetch and fan-out portion of the EMC's ASM chart. It also shows the turn-on sequence, such as it is. It merely amounts to starting up in state zero.

What is done in state zero is this. The Word Pointer is set to three in preparation for certain arithmetic machine-instructions, and the AE register is cleared. The DMP IDA/SET IL/SET N loads an instruction bit pattern into the Instruction Latch. Recall that, in the event of an interrupt or a non-EMC machine-instruction, all ones is loaded into the latch. The only time that a pattern other than all ones is loaded into the latch is when an actual EMC machine-instruction has been detected by Instruction Decode.

The transition from state one to state two is used to make the branch to the various major segments

of the ASM chart. The only instructions that are decoded in state one are "don't cares". It takes some time for the Instruction Latch and the asynchronous control lines to set up. The period between state one and state two is essentially a delay to allow this to occur.

Figure 14-4 is the segment of the ASM chart that corresponds to the CLR (Clear) machine-instruction. This machine-instruction zeros 1 to 16 consecutive words of memory, beginning with the address contained in the A register. The assembler encodes one less than the number of words to be cleared into the least four bits of the machine-instruction bit pattern. At the start of this segment, then, that less-one count is in the N Counter.

The first thing that is done is to read the starting address from

CLR SEGMENT OF THE EMC ASM CHART

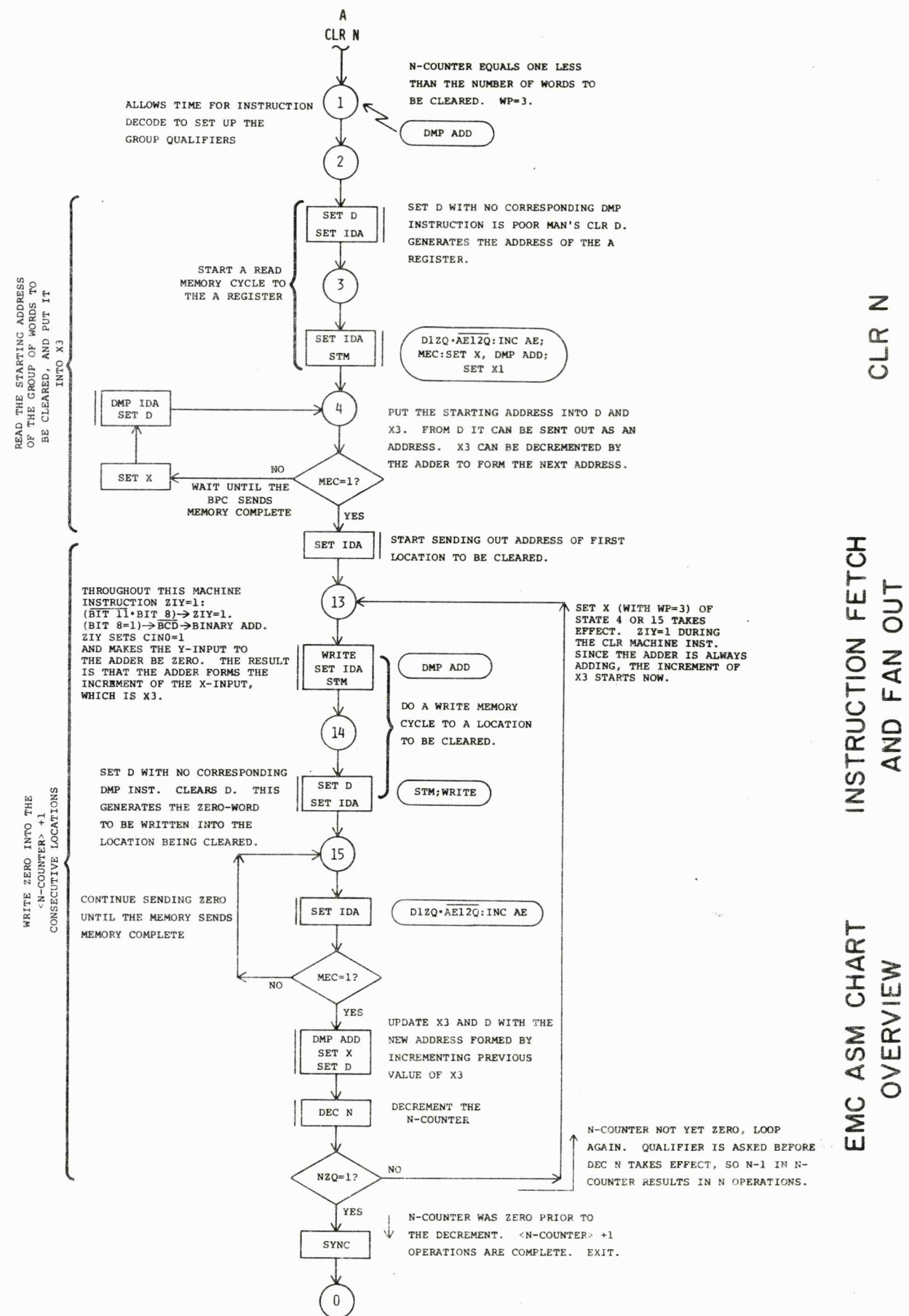


FIG 14-4

SECTION 14 (CONTINUED)

the A register and put that value into X3. Recall that the Word Pointer was set to 3 in state zero. Also, a SET D done with no corresponding DMP micro-instruction gets the all zeros that will be read into the locations to be cleared.

The main activity takes place in a loop that consists of write memory cycles. Each time through the loop the address in X3 will be incremented and the count in the N Counter decremented. The loop will be terminated when the qualifier NZQ is met, indicating that the N Counter has been counted down to zero. The decrement of the N Counter is easily obtained. It is done explicitly with the micro-instruction DEC N. The increment of X3 is a little bit more involved. The asynchronous control lines arrange that the instruction ZIY is issued. This causes one input to the Adder to be zero. They also arrange that the Adder feel a carry-in. Since the Word Pointer is three the other input to the Adder is X3. The sum of X3, 0 and a carry-in is $X3 + 1$.

Figure 14-5 is the ASM chart segment for the XFR (Transfer) machine-instruction. The purpose of this machine-instruction is to transfer one to sixteen words from one consecutive group of addresses to another. At the start of the routine the N Counter contains one less than the number of transfers to be performed.

The first thing that is done is to read from B the starting address of the destination locations. This is put into X2. The address of B is obtained with the micro-instruction DMP ONE. Then the A register is read to obtain the starting address of the source locations. This is put into X3. The address of A is obtained with a SET D which has no corresponding DMP micro-instruction. That sets D to all zeros, which is the address of A. All of this is done in a loop that uses manipulation of the Word Pointer to both select between X2 and X3 as well as to control the branching out of the loop.

The rest of the activity consists of a loop which is done one more time than the value of the count in the N Counter. That count is N-1; therefore, the activity is done N times. Each pass through the loop decrements the N Counter. The exit qualifier is NZQ (N Counter = 0). The decrement is issued in the same state that the qualifier is asked at the bottom of the loop. Due to the normal delays associated with any micro-instruction, however, that decrement N does not take place until after the qualifier has been asked and acted upon.

The working portion of the loop uses the contents of X3 as an address to read the next source location. The data obtained is temporarily saved in X1. After this, X3 is incremented. This increment is done in the same manner as in the CLR machine-instruction shown in Figure 14-4.

Next, a write memory cycle is started using the contents of X2 as the address and X1 as the data. After this, X2 is incremented and the loop qualifier tested.

Figure 14-6-1 is the ASM chart segment for the MRX and DRS machine-instructions. MRX shifts AR1 right the number of times indicated by the contents of the B register. B may be between zero and fifteen, inclusive. DRS shifts AR1 right one time. On the first shift the least four significant bits of A are shifted into D1 of AR1. On the last shift D12 is shifted into the least four significant bits of the A register. If only one shift is to be performed both these operations occur at the same time. Shifts that are neither the first nor last shift put zero into D1. This segment of flow charting has two entry points: one for MRX and one for DRS.

The entry point for MRX first reads the B register to determine the number of shifts to perform. That number is put into the N Counter. Also, the four least significant bits of the A register are read and placed into SE in preparation for the first shift. These two operations are done in a loop that uses manipulation of the Word Pointer for steering. Then a test is performed to see if an early exit is possible. That is, the number of shifts to be performed is tested to see if it is zero.

In the entry point for the DRS machine-instruction the N Counter already equals zero. SE is forced to be zero.

And now the two entry points converge. Next, AR1 is transferred to the X registers. This is done in a loop that uses the manipulation of the Word Pointer for both the designation of the various X's as well as for steering through the loop.

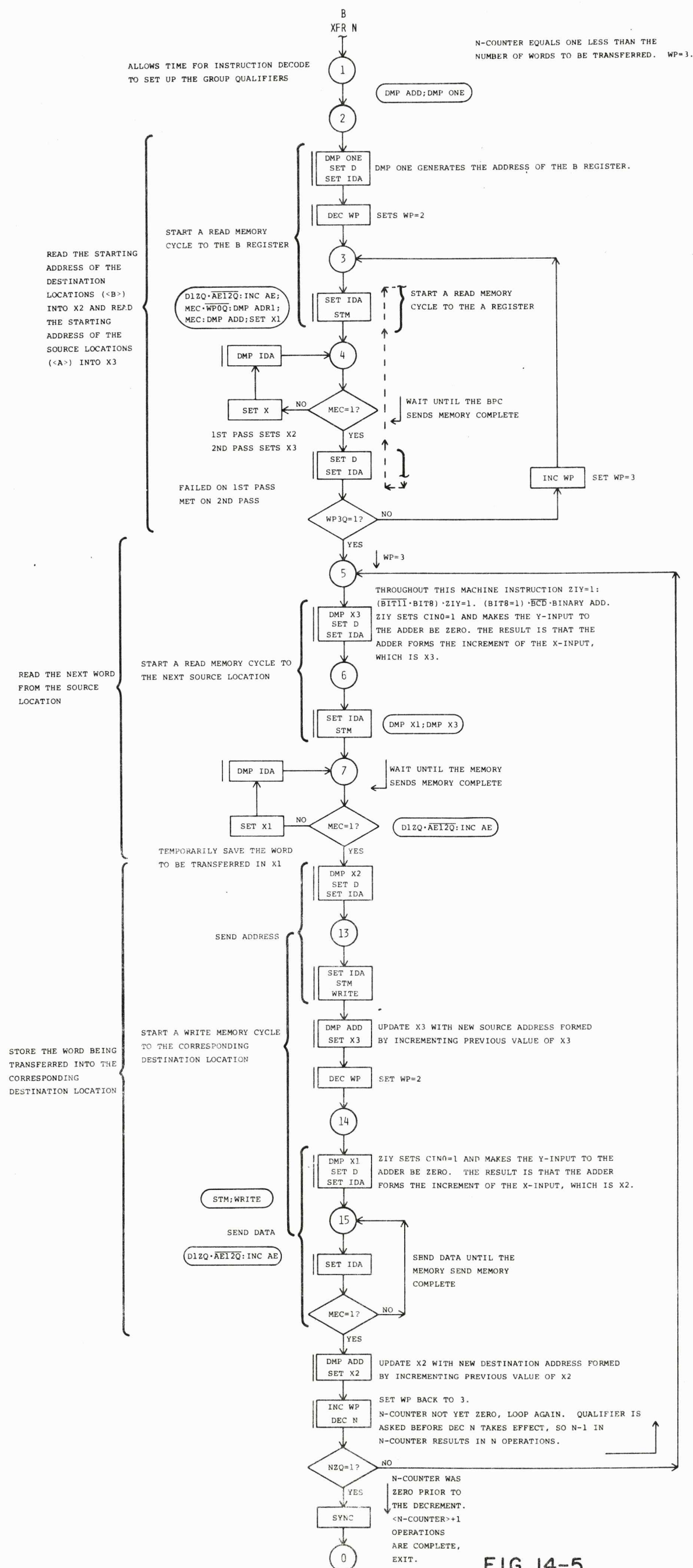
Now the X registers and SE are shifted the required number of digits. The asynchronous control lines control the direction of the shift. A shift of one bit will occur each time the micro-instruction SRE (Shift Register Enable) is issued. Each digit-shift requires a shift of four bits. Digit-shifts are performed by an outer loop around an inner loop that does four bit-shifts at a time. The inner loop uses the Word Pointer as a counter. The outer loop uses DEC N and NZQ for

steering.

Once the digits have been shifted the X registers are put back into AR1. The Word Pointer is once again used for steering. Then SE is put into A.

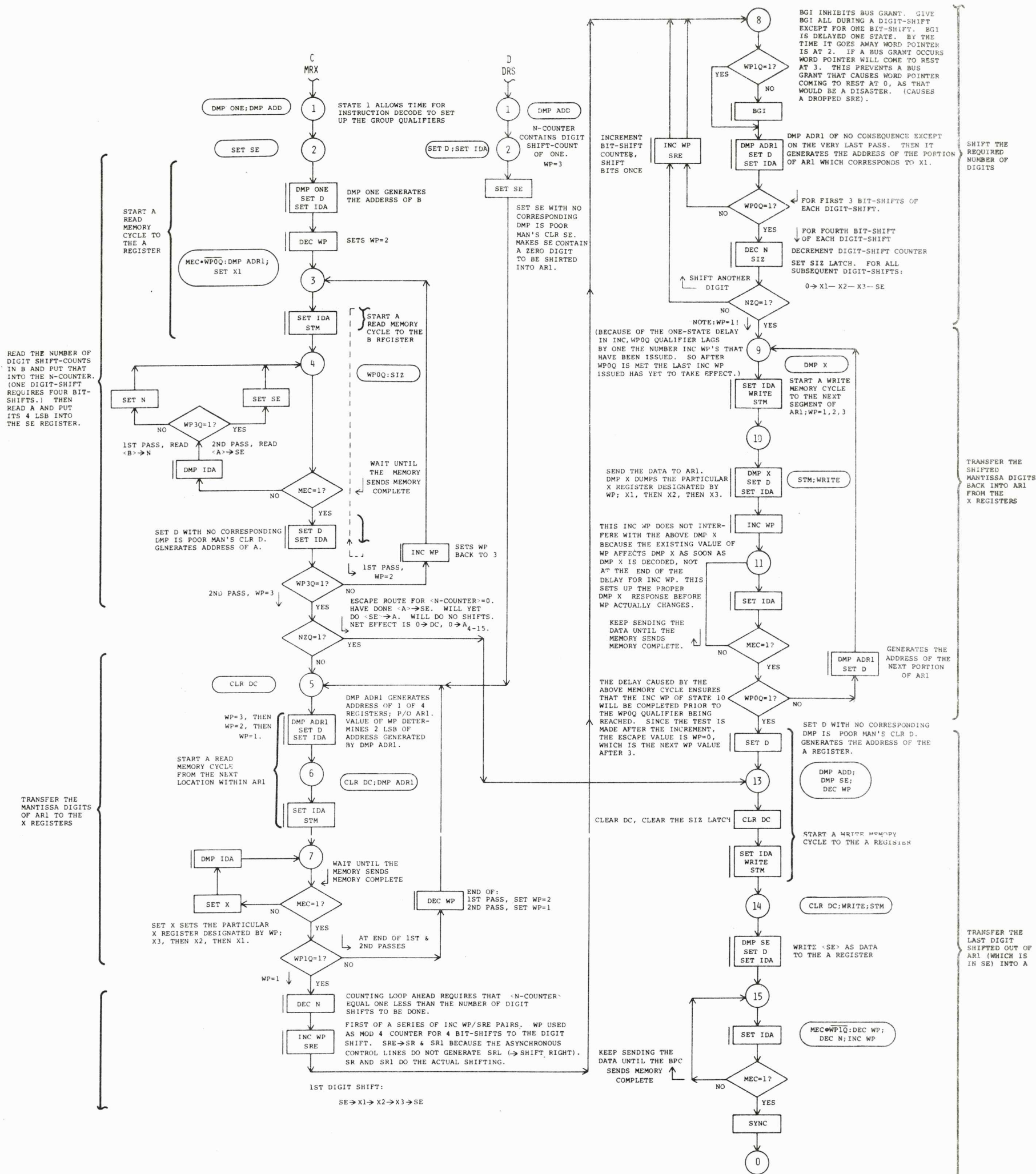
Figure 14-6-2 shows the timing relationships between digit shifting and allowable Bus Grant operations. There is some subtlety in the operation of this, involving a phenomenon known as "qualifier catch-up". The bit-shifting loop issues BGI's (Bus Grant Inhibits) to protect itself from the adverse affects of this phenomenon. Taken together, the flow chart and the timing diagram with its notes constitute a relatively self-explanatory description of what's going on. Don't be surprised if you don't get it the first time, however. A similar situation set the stage for a disastrous bug in the CMX portion of the ASM chart for the 15-bit version.

XFR SEGMENT OF THE EMC ASM CHART

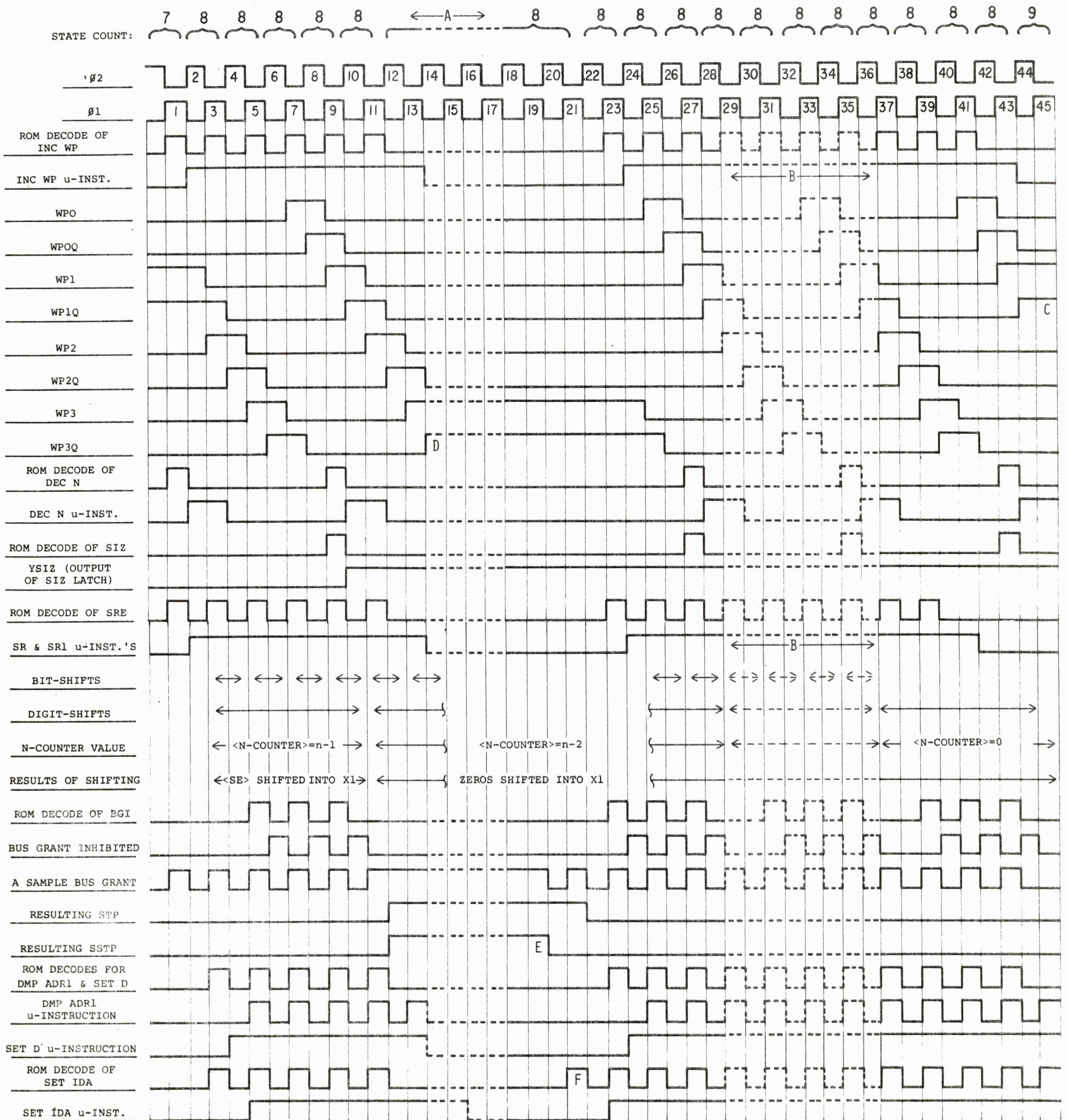




MRX AND DRS SEGMENTS OF THE EMC ASM CHART



TIMING OF DIGIT-SHIFT GEARWORKS COMPRISING STATES 7-8-9 OF THE MRX AND DRS PORTIONS OF THE EMC ASM CHART



NOTES

- ALL OF THIS PARTICULAR STATE 8 REPRESENTS AN EXTERNALLY ORIGINATED BUS REQUEST/ BUS GRANT OF INDEFINITE LENGTH.
- CLOCK-TIMES 29 THROUGH 36 (INCLUSIVE) REPRESENT ONE OF N-MANY DIGIT SHIFTS.
- THIS IS THE FINAL VALUE OF THE WORD POINTER AND IS THE VALUE IN EFFECT WHEN DMP ADRI IS GIVEN.
- NOTICE HOW THE WORD POINTER "CATCHES UP". IF A BUS GRANT WERE ALLOWED SUCH THAT THE CAUGHT-UP VALUE MET THE EXIT QUALIFIER, THE RESULT WOULD BE ONE LESS SRE THAN NECESSARY. HENCE BUS GRANTS ARE ALLOWED ONLY IN A WAY THAT PERMITS THE QUALIFIER LAG TO BE RE-ESTABLISHED WELL BEFORE THE EXIT QUALIFIER CAN BE MET.
- NOTICE THAT SSTP (SHORT STOP) ENDS ONE STATE SOONER THAN STP. THIS ALLOWS AN EXISTING SET IDA TO BE DECODED ONE EXTRA TIME PRIOR TO WHEN THE REST OF THE INSTRUCTIONS IN THAT STATE ARE. THIS IS TO RESTORE A SET IDA DIRECTLY AHEAD OF A SET IDA/STM IN THE EVENT THE INTERRUPTED STATE IS THE PART OF A MEMORY CYCLE THAT ISSUES THE SET IDA/STM.
- THIS IS THE EXTRA SET IDA DESCRIBED IN E.

FIG 14-6-2

MRX AND DRS

MRX, DRS &
BUS GRANT
OPERATION

SECTION 14 (CONTINUED)

Figure 14-7 is the segment of the ASM chart dealing with the MLY and MRY machine-instructions. MLY shifts AR2 left once, while MRY will shift it right the number of times indicated by the least four significant bits of the B register. Both instructions shift the four least significant bits of the A register into D1 on the first shift. Both instructions also shift D12 into the least four significant bits of the A register on the last shift. MLY does both of these things simultaneously. During intermediate shifts for MRY, zeros are shifted into D1. The two machine-instructions have different entry points.

The entry point for MLY leaves the Word Pointer set at three and sets the address of the A register into D. Also, on entry the N Counter already equals one. The N Counter indicates the number of shifts to be performed.

The entry point for MRY sets the Word Pointer down to two and puts the address of B into the D register.

At this point the two entry points converge onto a loop. The value of the Word Pointer on entry to the loop determines whether the loop will do one or two passes. For MLY there is one pass and A is put into SE. For MRY there are two passes. The first pass puts the contents of B into the N Counter (this sets up the number of shifts to be performed). The second pass puts the value of A into SE.

Once these preliminaries are done the number of shifts to be performed is checked with NZQ. If that number is zero then an early exit is possible.

The next thing that is done is to transfer the contents of the Y registers to the X registers. (Recall that AR2 is the collection of Y registers, but that only the X registers can be shifted.) What used to be in X is lost. Manipulation of the Word Pointer is used both to control the order in which registers are transferred and as a loop qualifier to exit from the loop which does the transferring.

Next, the digits are shifted. This is done in the same general manner as was described for MRX and DRS in Figure 14-6. The difference here is that the asynchronous control lines, in conjunction with which-ever instruction bit pattern is in Instruction Decode, control the direction of the shift. After this, the shifted digits are put back into the Y registers. The last digit shifted out is in the SE register; it is loaded into A.

The same general types of considerations concerning Bus Grants during digit-shifting also apply to MLY and MRY, as described earlier in connection with Figure 14-6.

Figure 14-8 is the ASM chart segment for the NRM (Normalize) machine-instruction. The purpose of NRM is to shift AR2 left until its D1 is not equal to zero. If the original D1 does not equal zero to begin with, no shifts occur. The number of shifts is returned as a binary number in B. If twelve shifts occur AR2 is zero; then no more shifts are done, but DC is set to a one.

On entry the AE register is already set to zero. This was done in state zero. AE is used as a counter to record the number of shifts performed.

First, a loop is used to transfer AR2 (the Y registers) to the X registers. The Word Pointer is used both to select the registers to be transferred, which is done one at a time, and also as the loop qualifier.

Next, SE and DC are both cleared. SE will be used as an intermediary register through which the zeros shifted out of the top of the X1 register wrap around into the bottom of the X3 register. If SE were not cleared it could induce a spurious digit onto the end of the mantissa being shifted.

The next point in the flow chart is the top of the shifting loop. At the very start of the loop D1 is checked to see if it is non-zero. As soon as that test is met the X registers are put back into the Y registers. Otherwise, the Word Pointer acts as a counter for a four bits at-a-time shift loop. After a shift of a single digit D1 is again checked to see if it is non-zero. If D1 remains zero, the process is repeated until either D1 fails to equal zero or twelve shifts with D1 equal to zero have occurred. In the latter case, DC is set to one, the shifting loop is exited and the restoring of the shifted zeros back into the Y register is bypassed. After all, they were all zeros to begin with and there is no point in reloading them with more zeros.

The last thing that is done is to transfer the count in the AE register into the B register. AE got incremented after each shift of

a digit, provided that the count was not already 12.

MLY AND MRY SEGMENTS OF THE EMC ASM CHART

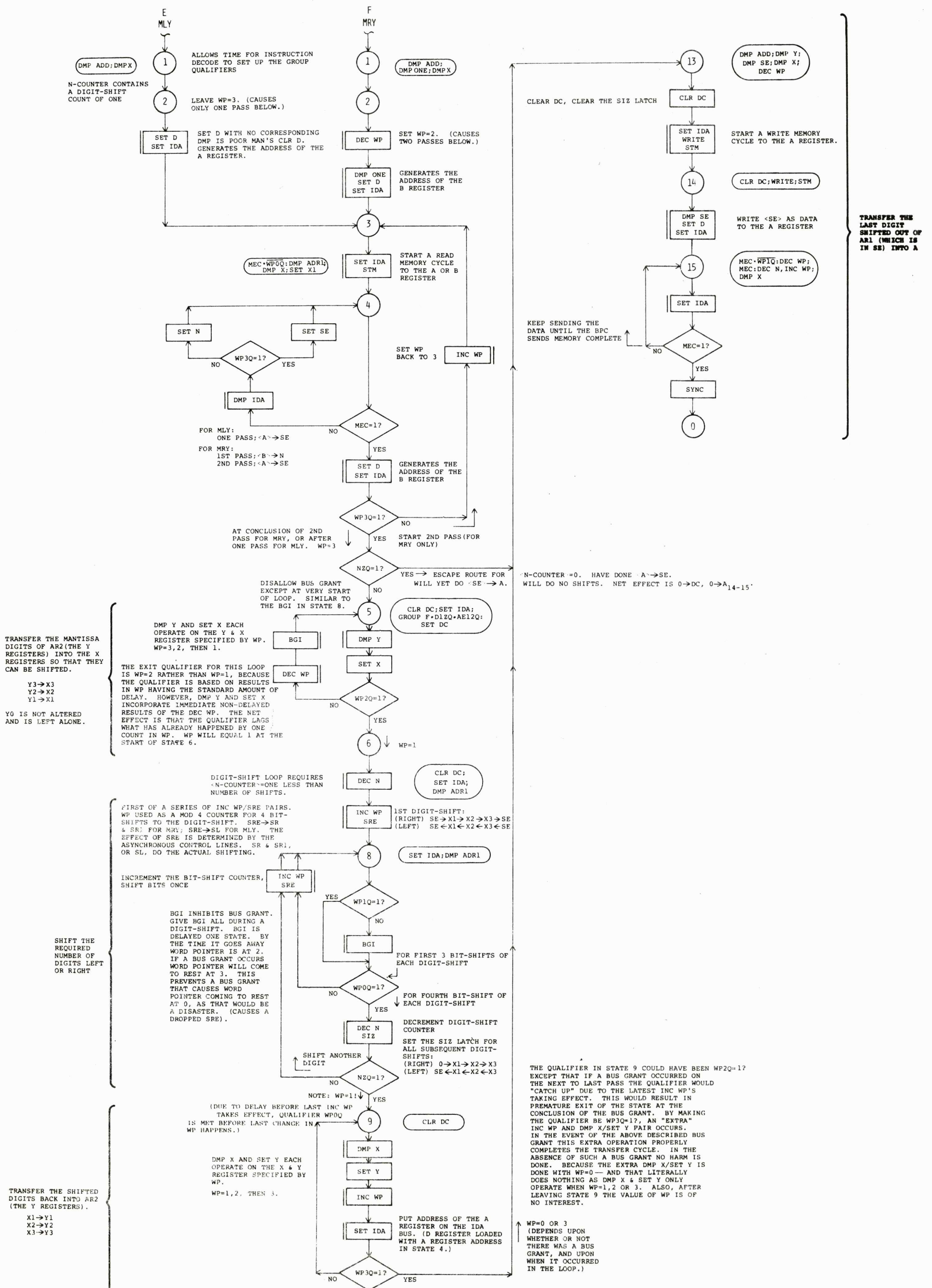


FIG 14-7

NRM SEGMENT OF THE EMC ASM CHART

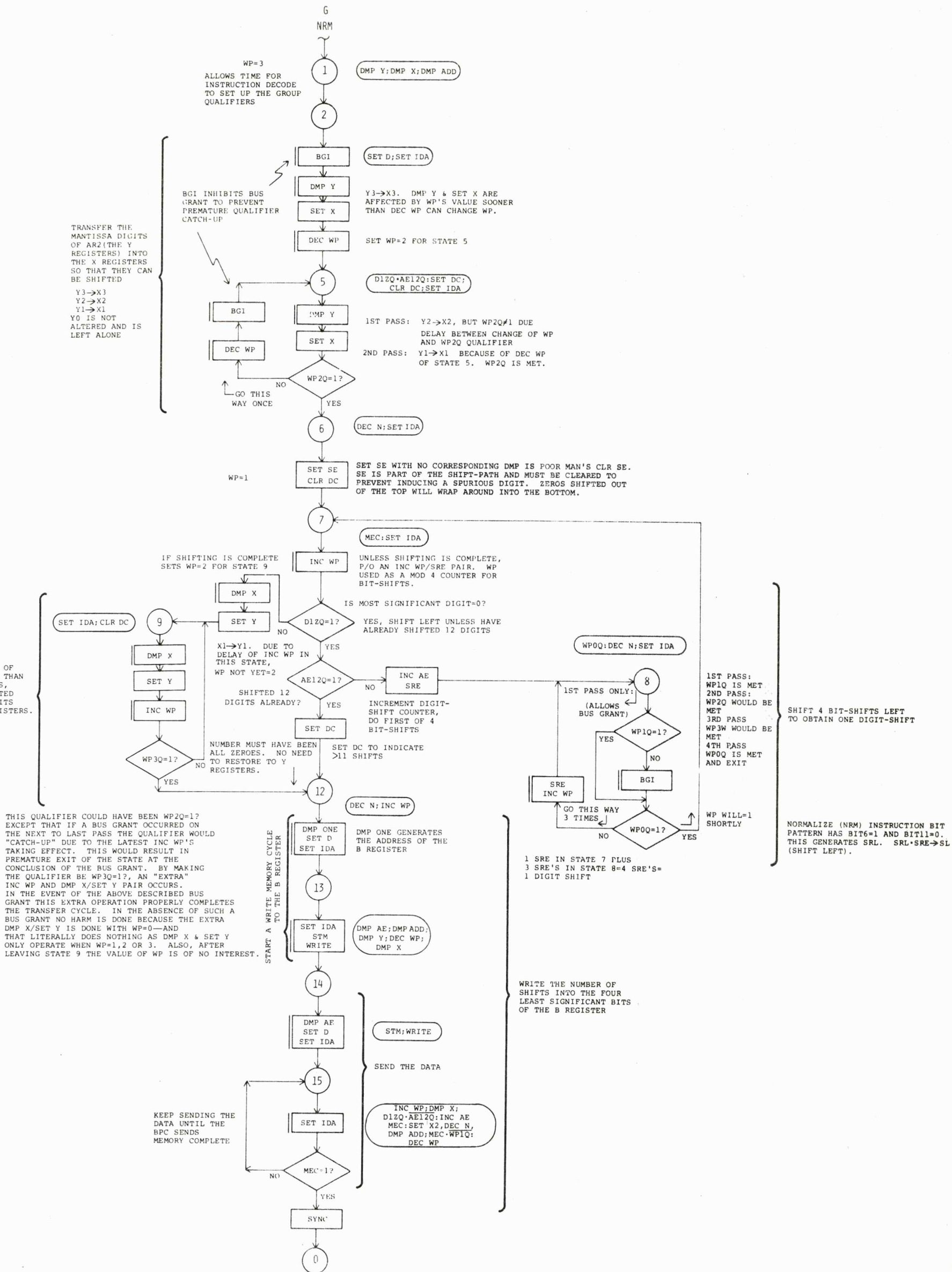


FIG 14-8

FXA PORTION OF THE EMC ASM CHART

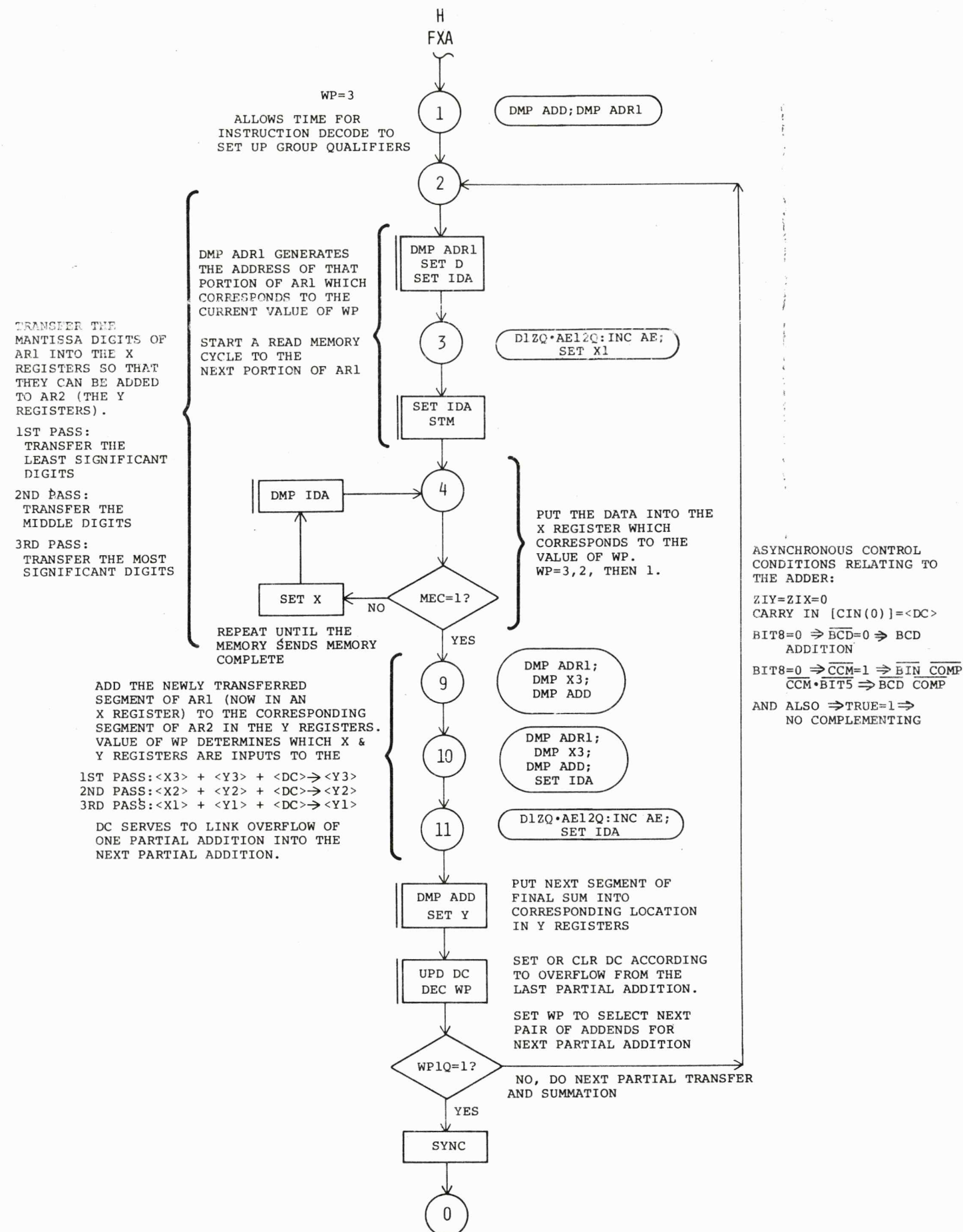


FIG-14-9

SECTION 14 (CONTINUED)

Figure 14-9 is the ASM chart segment of the FXA (Fixed-Point Addition) machine-instruction. The purpose of this machine-instruction is to add the mantissas of AR1 and AR2. Recall that AR1 is four consecutive words of memory outside the EMC, and that DMP ADRI in conjunction with the value of the Word Pointer generates the addresses of those words. AR2 is the Y registers. FXA ignores the exponent words and operates only on the mantissas.

The procedure is to set the Word Pointer to three and enter a loop that uses the Word Pointer to read a segment of AR1 into the corresponding X register. That X register is then added to its corresponding Y register. The result is left in that portion of the Y register. DC is part of the addition, and serves to link the carry-out from the addition of one segment to the addition of the next segment. After each segment the Word Pointer is tested to see if the last segment has just been added. If not, Decrement Word Pointer takes effect as the loop starts over. The order of addition of the segments is from least significant mantissa digit through the most significant mantissa digit.

On exit, DC represents overflow. The flow chart does not fool with DC — it is controlled strictly by the Adder.

FXA

NRM



Figure 14-10 is the MWA (Mantissa Word Add) segment of the EMC's ASM chart. MWA is a mantissa addition machine-instruction used primarily in rounding routines. The contents of the B register are taken as four BCD digits and added to AR2, in the D12 through D9 positions. DC is also added as a carry-in. On exit, DC reflects the existence of overflow from the addition.

The procedure is to read B and put it into X3. X1 and X2 are set to zeros. AR2, of course, is already in Y. From here on, the routine is a loop very similar to the one in FXA. The Word Pointer changes as the various segments are added one-at-a-time. Instruction Decode determines the BCD (as opposed to binary) nature of the addition, as well as the absence of complementing.

Figures 14-11-S and -F are the ASM chart segment for the CMX machine-instruction. The purpose of CMX is to form the BCD ten's complement of AR1. Recall that AR1 is four consecutive words of memory outside the EMC. Only the mantissa words are affected; the exponent word is left unchanged.

The basic procedure is to first save the current contents of Y1 - Y3 in X1 - X3, so that the Y inputs to the Adder are available. Then one at a time the various segments of AR1 are read into their associated Y register, complemented, and added to one or zero by the Adder. That forms the ten's complement. The output from the Adder is immediately written back into the associated segment of AR1.

AR1 is segmented into the Y registers and the Y registers taken in this order: Y3, Y2, Y1. DC is set prior to the complementing and addition of the first segment only. Thereafter, DC is used to link possible carry-outs from the addition of a segment into the addition of the next segment. Throughout all additions the asynchronous control lines force the IX Bus (the other input to the Adder) to zero, by issuing ZIX. After the addition is complete, which is

SECTION 14 (CONTINUED)

done in a Word Pointer controlled loop, the original contents of the Y registers are restored and the operation is then complete. There are some significant differences between the 15 and 16-bit versions; each version is described below.

The 16-bit version works as follows. On entry, it uses a Word Pointer controlled loop to transfer Y3 through Y1 into X3 through X1. The Word Pointer is then set back to 3 and the main complementing loop commences. This loop is also a Word Pointer controlled loop, and does each of the following things, in order, for each of these values of the Word Pointer: 3, 2, and 1.

- Reads the next portion of AR1 into the associated Y register.
- Starts a write memory cycle back to AR1.
- Forms the complement/sum of the segment.
- Finishes the write memory cycle to AR1. Restores the associated original Y value which has been saved in the associated X register.
- Tests the loop qualifier and decrements the Word Pointer. Loops if the loop is not yet finished.

The 15-bit version works as follows. On entry it uses a Word Pointer controlled loop to transfer Y3 through Y1 into X3 through X1. The Word Pointer is then set back to 3 and the main complementing loop commences. This loop is also a Word Pointer controlled loop, and does each of the following things, in order, for each of these values of the Word Pointer: 3, 2, and 1.

- Reads the next portion of AR1 into the associated Y register.
- Starts a write memory cycle back to AR1.
- Forms the complement/sum of the segment.
- Finishes the write memory cycle to AR1.
- Tests the loop qualifier and decrements the Word Pointer. Loops if the loop is not yet finished.

After the main complementing loop is finished a smaller loop transfers the contents of the X registers back into the Y registers. This restores the original Y register values.

The difference between the two versions is in the way the Y registers are restored with their original contents. In the 16-bit version this activity is integrated into the main complementing loop, while in the 15-bit version it is done by a separate loop, afterwards. The 15-bit version is afflicted with a serious bug, which we shall attempt to describe.

The process of restoring the original values of the Y registers actually begins at the bottom end of the last pass through the main complementing loop. At that point the Word Pointer equals one and a DMP X and SET Y occur. This is all done during the last pass through state fifteen. State fifteen also issues INC WP. This sets the Word Pointer to two for the first pass in the loop around state nine. Now, there was no chance of a Bus Grant interfering with the DMP X/SET Y of state fifteen, because a memory cycle was in progress. From here on however, it will be necessary to constantly consider the effects of a Bus Grant.

By the time state nine is reached the value of the Word Pointer is not yet two. This is because of the one state delay in series with all micro-instructions. However, by the time the DMP X and SET Y that are decoded in state nine are executed (that is, after their delays) the Word Pointer *will* be two. At the start of the second pass through state nine the qualifier WP2Q would be met if it were asked. *If it were* the exit qualifier another DMP X and SET Y *would still be issued* and they would affect X3 and Y3 because of the Increment Word Pointer issued during the first pass through state nine. However, that would be a *first class bug*. A Bus Grant at the start of the first pass through state nine would allow the INC WP in state fifteen to take effect before the loop qualifier in state nine was

asked. This is the phenomenon of qualifier catch-up. If the INC WP of state fifteen *does* have time to take effect before the qualifier of state nine is asked, and *if* that qualifier were WP2Q instead of WP3Q, that qualifier would be met the first time it is asked, instead of failed as intended. (Remember, the reason it's supposed to fail is because of the delay in getting the INC WP of state fifteen into effect.) Well, the engineering staff has been bitten by *this* kind of bug before, and the solution was to up the ante on the qualifier, by changing the qualifier in state nine to WP3Q. That solves this particular problem by always requiring an extra pass, anyway. Which, if qualifier catch-up does occur, allows the lag to be reintroduced and X3 and Y3 to be properly taken care of. In the absence of a Bus Grant at the start of state nine, the extra pass is done with the Word Pointer equal zero, which is accompanied by no SET or DMP, and is therefore harmless.

And, if the Bus Grant should occur at the start of the second pass through state nine, no harm would be done, because even though qualifier catch-up would occur, the second execution of the DMP X, SET Y would still occur, resulting in only the unnecessary third pass being chopped off, instead of the necessary second pass. Everything looked cool.

But alas, it was not to be. Suppose that a Bus Grant occurs at the start of the unnecessary third pass, then what? The INC WP that was issued during the second pass still takes effect, even though everything else on the ASM chart is "stopped". That is, one state later the qualifier WP3Q will go true. Now, recall that when STP is given it doesn't kill the ROM, per se. What it does is render most instructions non-decodable. *SYNC is not one of those instructions*. So, even though STP is given the ROM is trying to decode things, and SYNC obliges. The BPC understands a SYNC to mean that the next non-Bus Grant memory cycle is to be the instruction fetch. So far, so good. When the

Bus Grant is over, the BPC will initiate the instruction fetch. At the same time that the BPC starts the instruction fetch the EMC will decode the micro-instructions in state nine one last time. The DMP X/SET Y poses no problem. But observe that *there is also a don't-care decode of SET D/SET IDA in state nine!* Both the BPC and the EMC are doing a SET IDA. This garbles the address of the instruction fetch. *Finis!*

This was a super hard bug to find. Once the preliminary ground work had been laid to reveal the circumstances under which the problem happened, and which chip was generally at fault, subsequent investigation quickly revealed the machine-instruction and the very state in which the difficulty was occurring. From then on it was merely a problem of trying to figure out what the hell was wrong with the ASM chart. Even at that, it took several days more to understand the nature of the bug. Everybody thought that some wierd thing along the lines of earlier qualifier catch-up bugs was at work. All attention was directed to earlier events in the flow chart as the root cause.

The bug received a super fix, however. By moving the restoration of the Y registers back into the main complementing loop a whole state was saved. The phenomenon of qualifier catch-up does not affect that loop, because a portion of the relevant operations are done under the shadow of a memory cycle and another portion are done such that there is always enough delay for things to level out, anyway. Thus, there are no back-to-back increments with SET's and DMP's in a tight loop where the qualifier lags.

MWA

CMX PORTION OF THE EMC ASM CHART

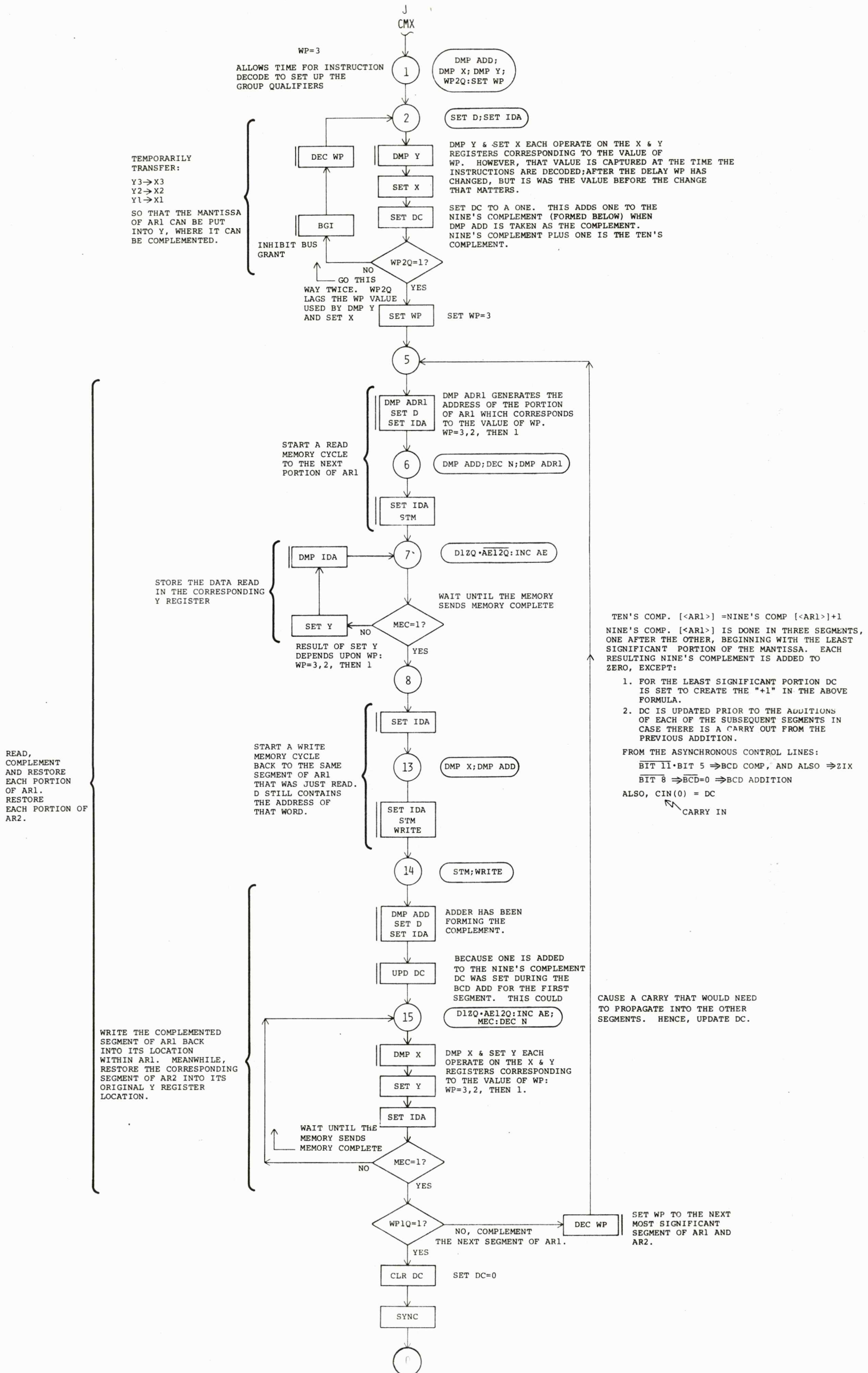


FIG 14-II-S

CMX SEGMENT OF THE EMC ASM CHART

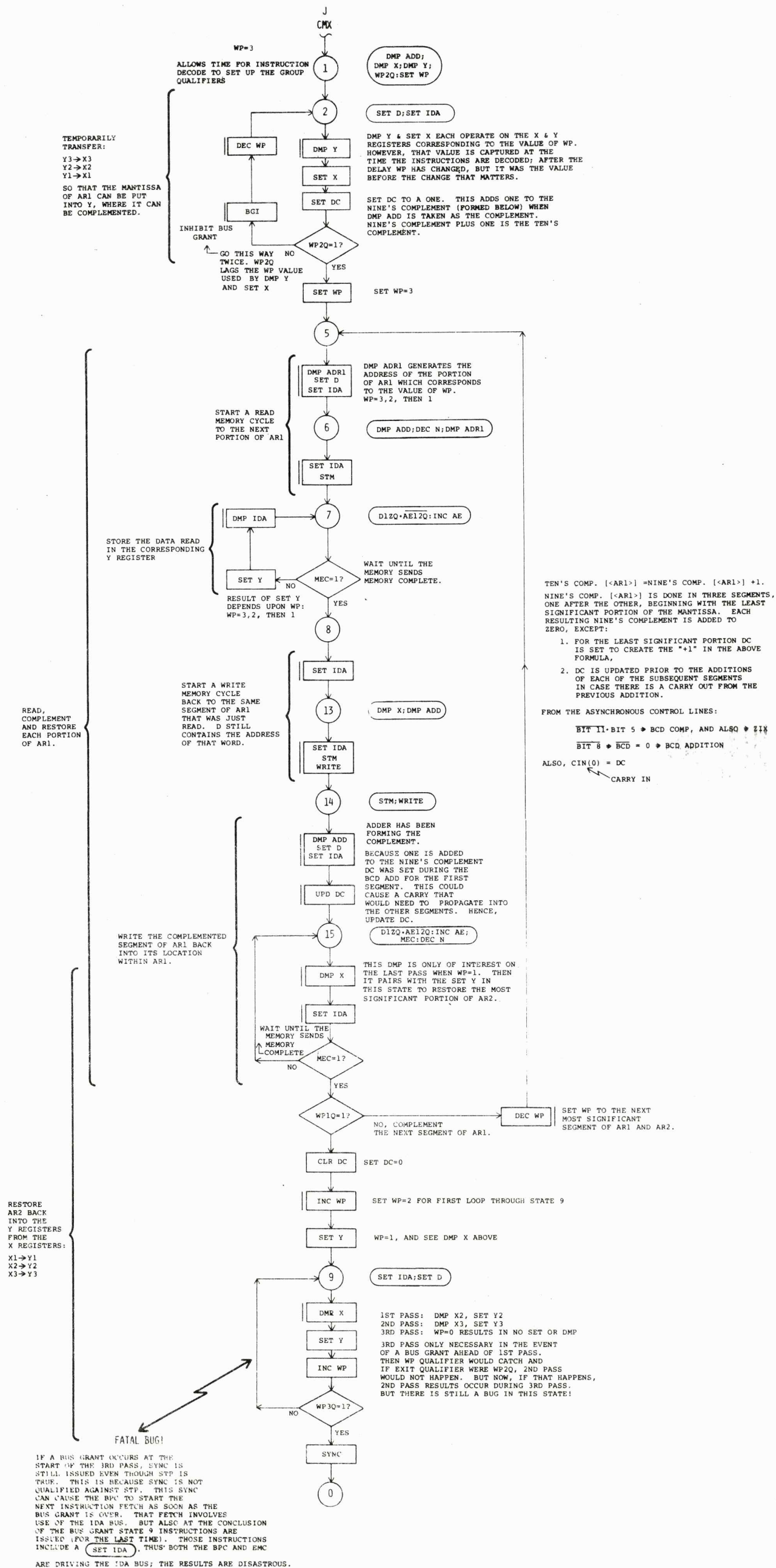


FIG 14-11-F



CMY SEGMENT OF THE EMC ASM CHART

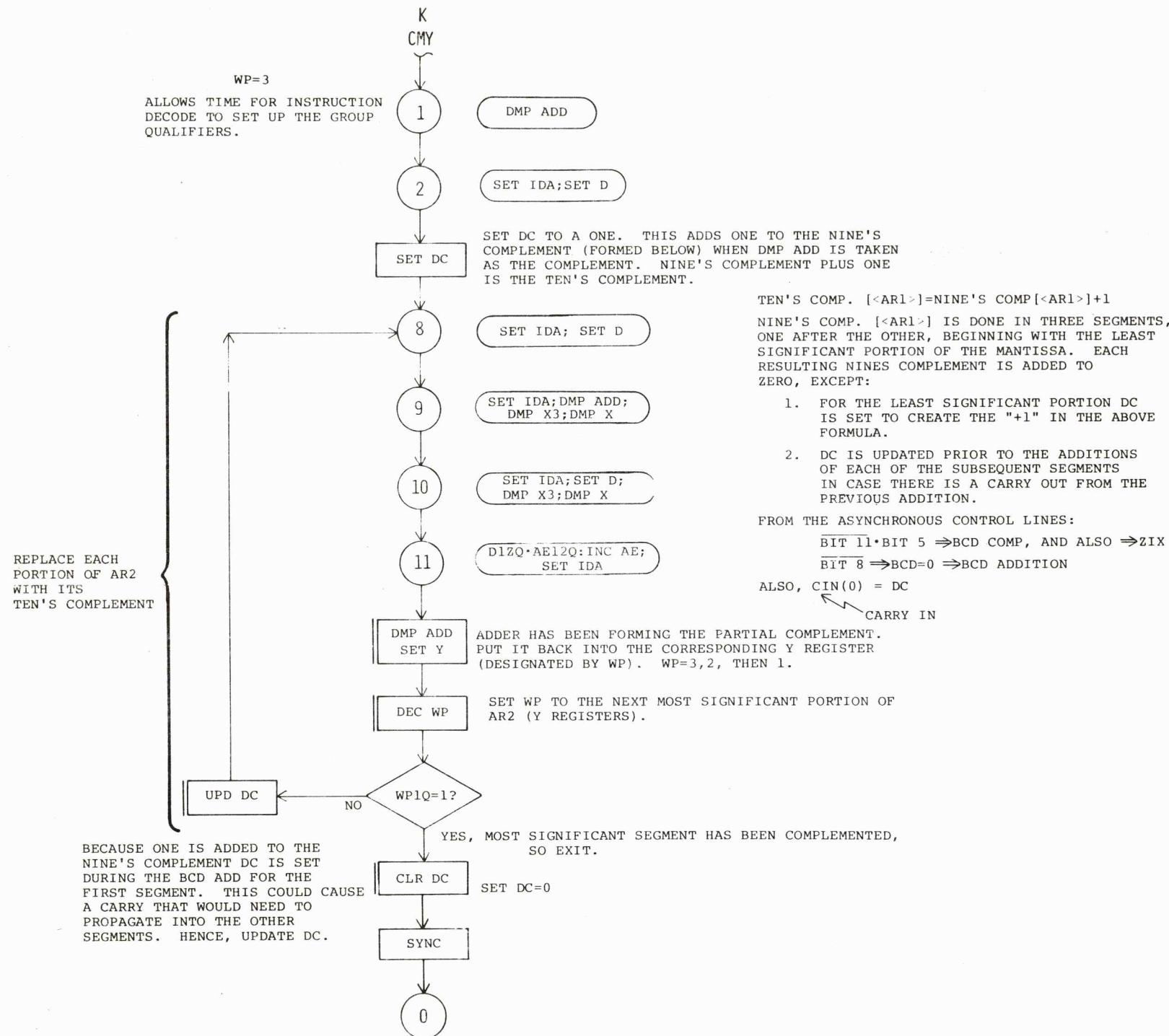


FIG 14-12

SECTION 14 (CONTINUED)

Figure 14-12 is the ASM chart segment for the CMY machine-instruction. Its purpose is to form the BCD ten's complement of AR2, which is the Y registers.

The method used is a subset of the activity performed by CMX. But in this instance the Y registers themselves are to be acted upon. They don't need to be saved and later re-

stored, nor does any external memory need to be referenced.

As before, the procedure is to add the nine's complement of Y3, a SET DC and 0 from the IX Bus. DC receives any carry-out. Then the nine's complement of Y2, 0 and DC are added, with the carry-out going into DC. Finally, the nine's complement of Y1, 0 and DC are added. This

activity is performed in a Word Pointer controlled loop.

Figure 14-13-1 is the flow chart for the FMP (Fast Multiply) and FDV (Fast Divide) machine-instructions.

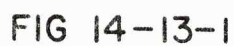
The purpose of FMP is to add the mantissa of AR1 to itself the number of times indicated by the least four significant bits of the B register. The result accumulates in AR2. The number of overflows is returned in the least four significant bits of the A register. By itself, FMP does not perform a complete floating point BCD multiplication. FMP is a useful machine-instruction that is used by software routines that do implement all the attributes of a floating point BCD multiply. Such routines involve additional mantissa shifts and exponent manipulation that are not provided by FMP.

The purpose of FDV is to add the mantissas of AR1 and AR2 together until the first decimal overflow occurs. The result of these additions accumulates in AR2. However, the additions are performed not to obtain the resulting sum, but to determine the number of such additions. The number of additions prior to the overflow is returned in the least four significant bits of B. By itself, FDV is not a complete divide instruction. FDV is an useful instruction used by floating point BCD division routines and by square root routines. It facilitates the repeated subtraction of AR1 from the combination of A and AR2. AR2 must be complemented prior to the execution of FDV. The using routine must also provide mantissa shifts and exponent manipulation not performed by FDV.

The N-Mos II Processor book contains extensive explanations of the application of FMP and FDV. Before attempting to understand the flow charts for FMP and FDV it's a good idea to first become familiar with the substance of those explanations.

The flow chart has two entry points; one for FMP and one for FDV. Below the entry points are both common segments and segments unique to each machine-instruction. The group qualifiers provided by the Instruction Latch are the basis for branching between the unique segments. We will treat the flow chart as if it were drawn twice — once for FMP sans FDV

211



SINGLE DIGIT DEMONSTRATION OF THE COMPLEMENT ARITHMETIC USED BY FDV

SUPPOSE: AR1 HAS ONE DIGIT,
AR2 HAS ONE DIGIT,
AND THAT EACH HAS THE SAME SIGN (ASSUME THEM BOTH TO
BE POSITIVE).

CONSIDER AR2-AR1 BY COMPLEMENTING AR2 AND ADDING TO AR1.

$$(1) \quad (10-AR2)+AR1=10-(AR2-AR1)$$

$$(2) \quad =10+(AR1-AR2)$$

i. NOW SUPPOSE $AR2 > AR1$, SO THAT THE SUBTRACTION WILL BE SUCCESSFUL. THEN IN (1), $AR2-AR1$ IS BOTH POSITIVE AND LESS THAN 10, SO THAT $(10-(AR2-AR1))$ IS ANOTHER POSITIVE DIGIT LESS THAN 10, (i.e., THERE IS NO OVERFLOW). IN FACT, IT IS THE COMPLEMENT OF $AR2-AR1$.

HENCE:

$$(3) \quad AR2-AR1=(10-AR2)+AR1=10-(AR2-AR1)=0\overline{X}$$

WHERE X IS THE SINGLE DIGIT: $X=AR2-AR1$

ii. NOW SUPPOSE $AR2 < AR1$, SO THAT THE SUBTRACTION WILL BE UNSUCCESSFUL. THEN IN (2), $9 \geq (AR1-AR2) > 0$ AND $10+(AR1-AR2)$ IS GREATER THAN TEN (i.e., THERE IS OVERFLOW).

HENCE:

$$(4) \quad AR2-AR1=10+(AR1-AR2)=1\overline{X}$$

WHERE X IS THE SINGLE DIGIT: $AR1-AR2$

THE 1 IN FRONT OF THE X SHOULD BE DROPPED, AS THE ABSOLUTE VALUE OF THE ANSWER IS, AFTER ALL, LESS THAN 10. (REMEMBER, WE ASSUME AR1 AND AR2 HAVE THE SAME SIGN.) THE X IS THE DIGIT ASSOCIATED WITH THE NEGATIVE RESULT, SINCE $X=AR1-AR2=-(AR2-AR1)$.

iii. FINALLY, SUPPOSE $AR2=AR1$. THE SUBTRACTION WILL BE SUCCESSFUL, BUT OVERFLOW WILL STILL OCCUR. EQUATION (1) REDUCES TO $10-0$, AND EQUATION (2) REDUCES TO $10+0$. IN EACH CASE THERE IS OVERFLOW ACCOMPANIED BY AN "ANSWER DIGIT" OF ZERO.

SECTION 14 (CONTINUED)

and once for FDV sans FMP. Upon entry to each routine, AE is clear; a result of a CLR AE in state zero during the instruction fetch sequence.

FMP

Prior to the first use of FMP, AR2 will be set to zero. AR2 will be used to accumulate the partial products as they are formed. AR2 is also shifted right inbetween each use of FMP. FMP is given once for each multiplier digit processed.

The basic procedure is to add AR1 (the multiplicand) to AR2 the number of times indicated by the least four significant bits of B. The number of overflows is recorded in AE, and will later be put into A. AR2 will either be zero (to begin with) or contain the shifted partial product from the previous use of FMP. The least four bits of B will contain the successive multiplier digits.

On entry the first thing that is done is to transfer the multiplier digit from B to the N-Counter. Then the X registers are each set to zero by the Word Pointer controlled loop of state four. This is done in anticipation of the multiplier digit being zero. Normally it is not, and AR1 will be transferred to the X registers and added to AR2 the appropriate number of times. However, the routine is not laid out to add zero times. The main addition loop assumes the N-Counter is one less than the number of additions to be performed. If the multiplier digit is not zero it is decremented once before entering that loop to preserve this convention. If the multiplier digit starts out as zero the decrement mentioned in the previous sentence is bypassed and the main addition loop entered. It will then add the X registers to AR2 one time. By setting the X registers to zero when the multiplier digit is zero this addition is rendered harmless.

In the event that the multiplier digit is non-zero AR1 is transferred to the X registers by a Word Pointer controlled loop in states five, six and seven. At the conclusion of that loop the multiplier digit is

decremented to match the exit qualifier conventions of the main addition loop. Also, the Word Pointer is set back to three in preparation for entering the next loop.

The main addition loop commences at state eight. States eight, nine, ten and eleven are a Word Pointer controlled loop that successively adds the segments of the X registers (which are either zero or represent AR1) to the segments of AR2 (the Y registers). DC is used to link any carry-outs from the addition of one segment into the carry-in for the addition of the next segment. One pass through this loop accounts for activity corresponding to an original count of one in B. If that count was higher the loop of states eight through eleven must be repeated until it has been done the correct number of times. Each time the loop is finished the Word Pointer is set back to three in anticipation of the next use of the loop.

State twelve accomplishes this repeated use of the states eight through eleven loop by decrementing the N-Counter and checking to see if its undecrement value was zero. If the value was not zero the addition loop is repeated. If the value of the N-Counter was zero the loop is exited. In either case, AE is incremented if the addition loop produced an overflow. Thus, AE constitutes the most significant digit of the accumulation in AR2. When AR2 is right-shifted by the using routine, that digit will be put into the D1 position of AR2. That occurs as follows. On exit from the addition loop the count in AE is stored in the least four significant bits of A. Recall that the right shift machine-instruction MRY shifts A into the D1 position of AR2.

FDV

Upon arrival at the entry point for FDV the N-Counter equals one. This is a result of the SET N in state zero of the instruction fetch sequence and of the choice of bit pattern for the FDV machine-instruction. The sole purpose of the N-Counter's being one is to ensure that

FIG 14-13-2

MPY SEGMENT OF THE EMC ASM CHART

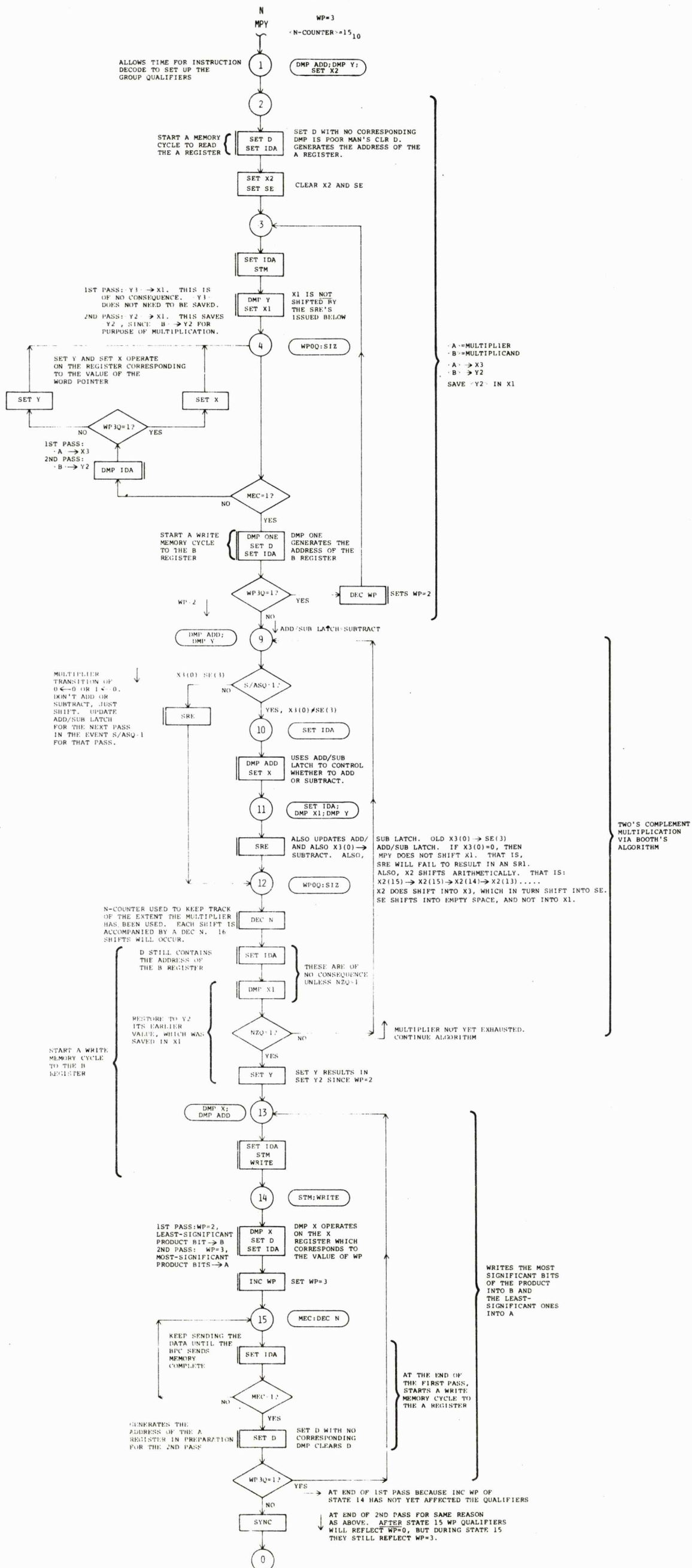
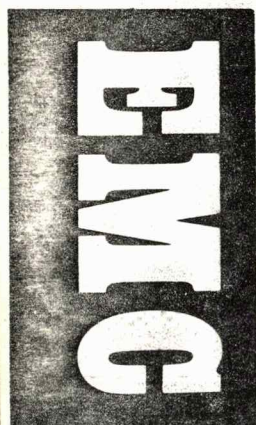
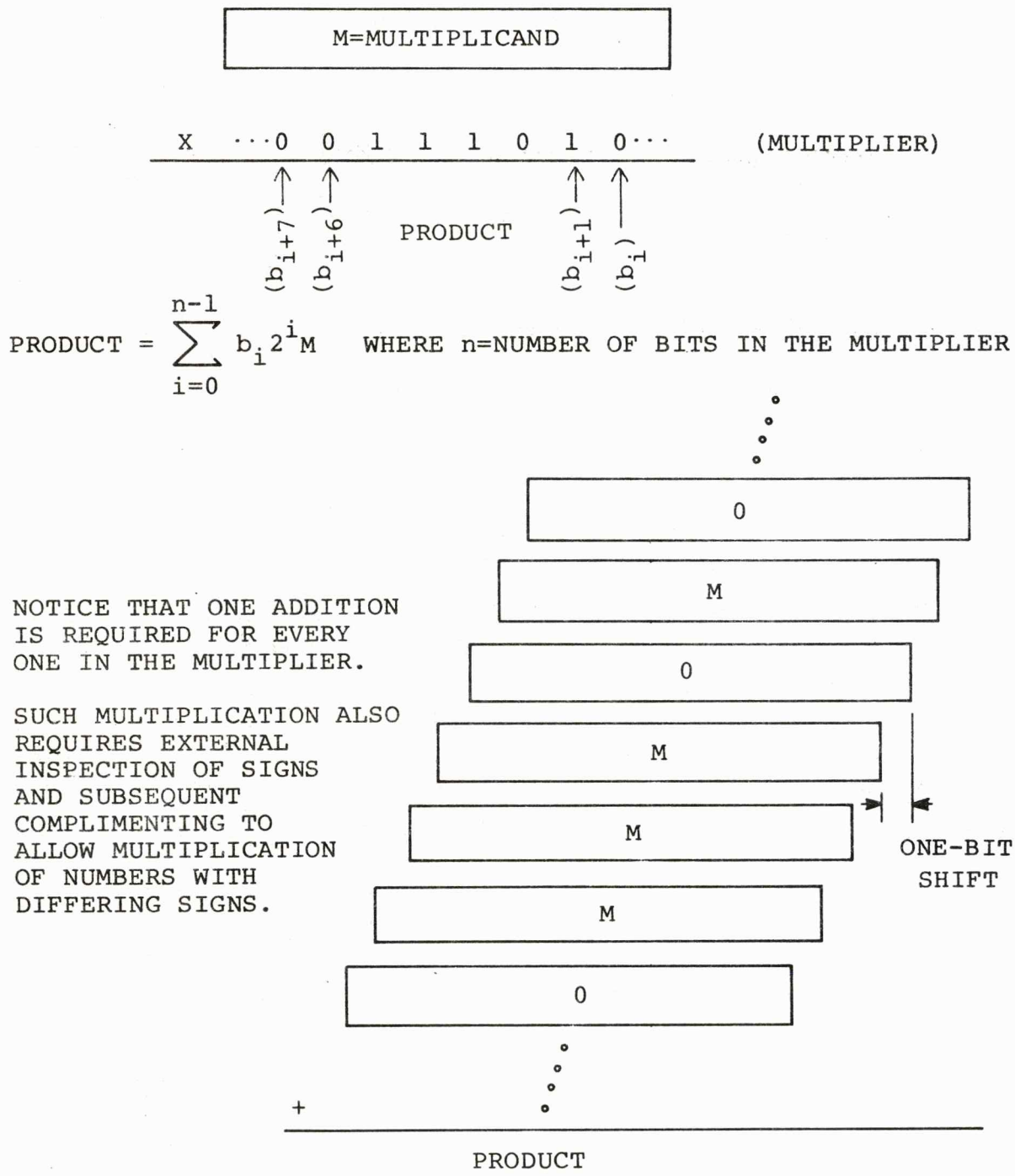


FIG 14-14-1



THE PRINCIPLE OF "STANDARD" BINARY MULTIPLICATION



NOTICE THAT ONE ADDITION IS REQUIRED FOR EVERY ONE IN THE MULTIPLIER.

SUCH MULTIPLICATION ALSO REQUIRES EXTERNAL INSPECTION OF SIGNS AND SUBSEQUENT COMPLIMENTING TO ALLOW MULTIPLICATION OF NUMBERS WITH DIFFERING SIGNS.

FIG 14-14-2

SECTION 14 (CONTINUED)

the early exit mechanism for multiplication by zero will not be met. Otherwise, the N-Counter plays no role in the operation of FDV.

The entry point for FDV leads directly to the Word Pointer controlled loop of states five, six and seven that transfers AR1 to the X registers.

Next, the Word Pointer is set back to three and the main addition loop of states eight, nine, ten and eleven is entered. This loop adds AR1 and (the previously complemented) AR2 together, using DC to perform carry-out/carry-in linkage between segments. At the conclusion of each such complete addition the Word Pointer is set back to three in anticipation of another use of that loop.

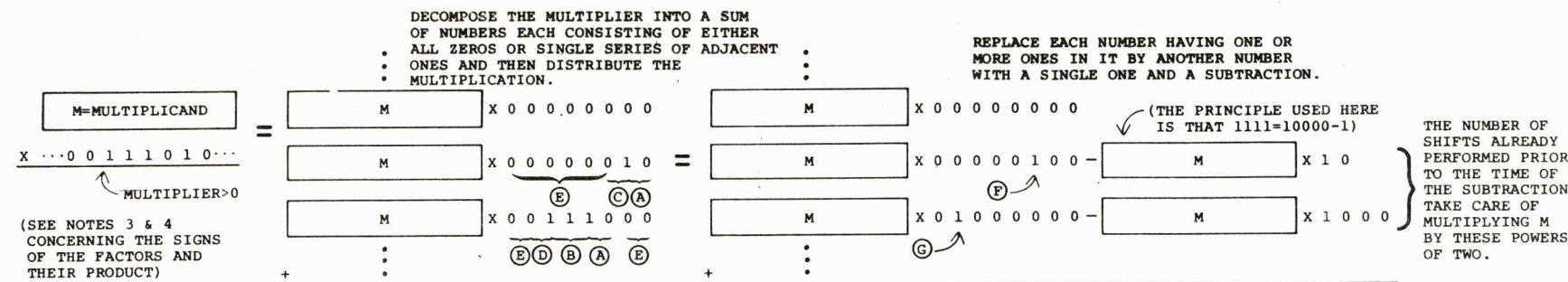
Next, DC is cleared while its uncleared value is used to control the exit from the main addition loop. As shown by arithmetic demonstration accompanying the flow chart (in Figure 14-13-2) the first "unsuccessful subtraction" is indicated by an overflow (same as carry-out) from the addition. That is, by the value of DC after the addition of the last segments of AR1 and AR2. If there was no overflow the subtraction was successful, and AE is incremented in honor of this fact. Then the addition is repeated. When the overflow finally does occur AE remains untouched while a branch is made to the actual exit segment. It writes the accumulated number of overflows in AE into the B register.

Figures 14-14-1 through -6 concern the MPY machine-instruction. The purpose of MPY is to provide a signed two's complement binary multiply using Booth's algorithm. Prior to using MPY the A register is loaded with the multiplier and B is loaded with the multiplicand. The 32-bit result is returned in the B and A registers, with B receiving the most significant bits and A receiving the least significant bits.

Figure 14-14-1 is the ASM chart segment for the MPY machine-instruction. It would be impossible to figure out Booth's algorithm merely from looking at this flow chart, or to understand the flow chart without having some notion of how Booth's algorithm works. Other drawings in this section explain how Booth's algorithm works.

Figure 14-14-2 illustrates, for the purpose of comparison and as a place to start, the principles of "standard" binary multiplication. Nothing strange about that.

OPERATION OF BOOTH'S ALGORITHM WHEN THE MULTIPLIER IS POSITIVE, OR WHEN ONE OF THE FACTORS IS ZERO.

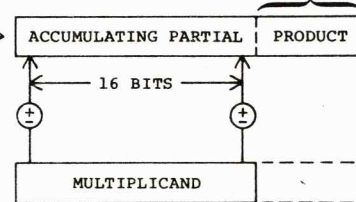


HOW THE MULTIPLIER IS USED AS IT IS SCANNED, RIGHT-TO-LEFT, ONE BIT AT A TIME:

- (A) A ZERO-TO-ONE TRANSITION REQUIRES AN IMMEDIATE SUBTRACTION, FOLLOWED BY A SHIFT.
- (B) SUBSEQUENT ONE-TO-ONE TRANSITIONS THEN REQUIRE ONLY WHAT WOULD NORMALLY BE REQUIRED FOR ZERO-TO-ZERO TRANSITIONS, I.E., ONE SHIFT EACH.
- (C) (D) THESE ONE-TO-ZERO TRANSITIONS CORRESPOND TO ONES (F) AND (G), RESPECTIVELY, AND EACH REQUIRES AN ADDITION, FOLLOWED BY A SHIFT.
- (E) A ZERO-TO-ZERO TRANSITION REQUIRES ONLY A SHIFT.

SUCCESSIVE ADDITIONS AND SUBTRACTIONS OF INCREASING POWERS-OF-TWO TIMES M ARE ACHIEVED BY SHIFTING THE ACCUMULATION TO THE RIGHT.

THIS IS ZERO PRIOR TO ANY ADDITIONS OR SUBTRACTIONS



SINCE NO OTHER USE IS MADE OF THE MULTIPLIER, IT CAN BE RIGHT-SHIFTED INTO A BIT TRANSITION MECHANISM, AND THE PORTION ALREADY USED THROWN AWAY.

*NOT TRUE IN 16-BIT COMPLEMENT ARITHMETIC IF THE MULTIPLICAND IS 1 000 000 000 000 000 (+32768). THE ALGORITHM FAILS WITH THAT MULTIPLICAND FOR THIS REASON. SEE THE BUG DESCRIPTION AT THE END OF THIS SECTION.

FIG 14-14-3

OPERATION OF BOOTH'S ALGORITHM WHEN THE MULTIPLIER IS NEGATIVE

1. IN THE EVENT THAT THE MULTIPLIER IS NEGATIVE, THE SIGN OF THE PRODUCT IS OPPOSITE THE SIGN OF MULTIPLICAND. WE SHALL DIVIDE THE POSSIBLE INSTANCES OF MULTIPLYING BY A NEGATIVE MULTIPLIER INTO THREE CATEGORIES AND SHOW THAT PROPER RESULTS ARE OBTAINED IN EACH CASE.

CASE I PRODUCT = -1.M

LET M=MULTIPLICAND
LET MULTIPLIER = -1 = 1111111111111111
← 16 BITS →

THIS CASE WORKS BECAUSE THERE IS AN IMMEDIATE ZERO-TO-ONE TRANSITION, CAUSING A SUBTRACTION FROM ZERO (WHICH GIVES THE PARTIAL PRODUCT A SIGN OPPOSITE THAT OF THE MULTIPLICAND). BUT SINCE THE REST OF THE MULTIPLIER IS ALL ONES, ONLY ARITHMETIC SHIFTS FOLLOW THIS SUBTRACTION.

THE COMPLEMENTED MULTIPLICAND IS SHIFTED TO FAR RIGHT OF THE 32-BIT ANSWER, THUS ITS MAGNITUDE (ABSOLUTE VALUE) REMAINS UNCHANGED, AND SINCE THE SHIFTS ARE ARITHMETIC SHIFTS, THE SIGN IS PRESERVED.

CASE II PRODUCT = -2^P.M

LET M=MULTIPLICAND
LET MULTIPLIER = -2^P = 111100 ... 0
← 16 BITS →

IN THIS CASE THERE ARE P LEADING ZERO-TO-ZERO TRANSITIONS, EACH OF WHICH SHIFTS A PARTIAL PRODUCT WHICH IS ZERO, AS NOTHING HAS BEEN ACCUMULATED YET. SO THOSE SHIFTS HAVE ABSOLUTELY NO EFFECT.

THE SINGLE ZERO-TO-ONE TRANSITION CAUSES A SUBTRACTION FROM ZERO, WHICH ESTABLISHES THE SIGN OF THE PRODUCT AS OPPOSITE THAT OF THE MULTIPLICAND. THE REMAINING ONES IN THE MULTIPLIER CAUSE 16-P ARITHMETIC SHIFTS, WHICH PRESERVE THE SIGN. BUT THESE SHIFTS FALL P SHIFTS SHORT OF FULLY SHIFTING THE COMPLEMENTED MULTIPLICAND TO THE RIGHT IN THE 32-BIT ANSWER SPACE. THIS IS AN EFFECTIVE LEFT-SHIFT OF P PLACES IN THAT 32-BIT SPACE. HENCE THE PRODUCT IS THE COMPLEMENT OF THE MULTIPLICAND, MULTIPLIED BY 2^P.

CASE III PRODUCT = -Y.M

LET M=MULTIPLICAND
LET -Y REPRESENT A NEGATIVE NUMBER DIFFERENT THAN -1 OR THE NEGATIVE OF A POWER OF 2:

-Y ≠ -1

-Y ≠ -2^K

THEN -Y CAN BE DECOMPOSED INTO THE SUM OF SOME X > 0 AND -2^P FOR SOME P:

$$-Y = 111010110 \dots = 111000000 \dots - 2^P + 010110 \dots = X$$

$$111010110 \dots = -Y$$

THEN, $-Y.M = (-2^P + X).M = -2^P.M + X.M$

AS THE MULTIPLIER IS SCANNED, X.M IS FORMED IN THE FASHION FOR POSITIVE MULTIPLIERS. THEN THE PRODUCT FOR -2^P.M IS ACCUMULATED TO IT. THE PROCEDURE OF THE ALGORITHM IS SUCH THAT THE FORMING OF X.M IS INDEPENDENT OF, AND DOES NOT INTERFERE WITH, THE SUBSEQUENT FORMATION OF -2^P.M. IT IS, SO TO SPEAK, AS IF THE FORMATION OF -2^P.M PICKS UP WHERE FORMING X.M LEAVES OFF. THE ONLY DIFFERENCE IS THAT IN THE FORMATION OF -2^P.M THE MULTIPLICAND IS NOT SUBTRACTED FROM ZERO, BUT FROM X.M. THE SIGN OF THE RESULT OF THAT SUBTRACTION WILL BE OPPOSITE THE SIGN OF X.M, SINCE 2^P.X.M. SINCE X.M HAS THE SIGN OF THE MULTIPLICAND, THIS MEANS THE FINAL PRODUCT HAS THE SIGN OPPOSITE THAT OF THE MULTIPLICAND, WHICH IS CORRECT.

*NOT TRUE IN 16-BIT COMPLEMENT ARITHMETIC IF THE MULTIPLICAND IS 1 000 000 000 000 000 (+32768). THE ALGORITHM FAILS WITH THAT MULTIPLICAND FOR THIS REASON. SEE THE BUG DESCRIPTION AT THE END OF THIS SECTION.

FIG 14-14-4

SECTION 14 (CONTINUED)

Figures 14-14-3 and -4 illustrate the principles of Booth's algorithm. Your author feels he cannot improve upon these and that they are best left to speak for themselves.

P/O BOOTH'S ALGORITHM

P/O BOOTH'S ALGORITHM

"STANDARD" BINARY MULTIPLICATION

BLOCK DIAGRAM OF THE HARDWARE CONTROLLED BY THE FLOW CHART WHICH DOES THE BOOTH'S MULTIPLY.

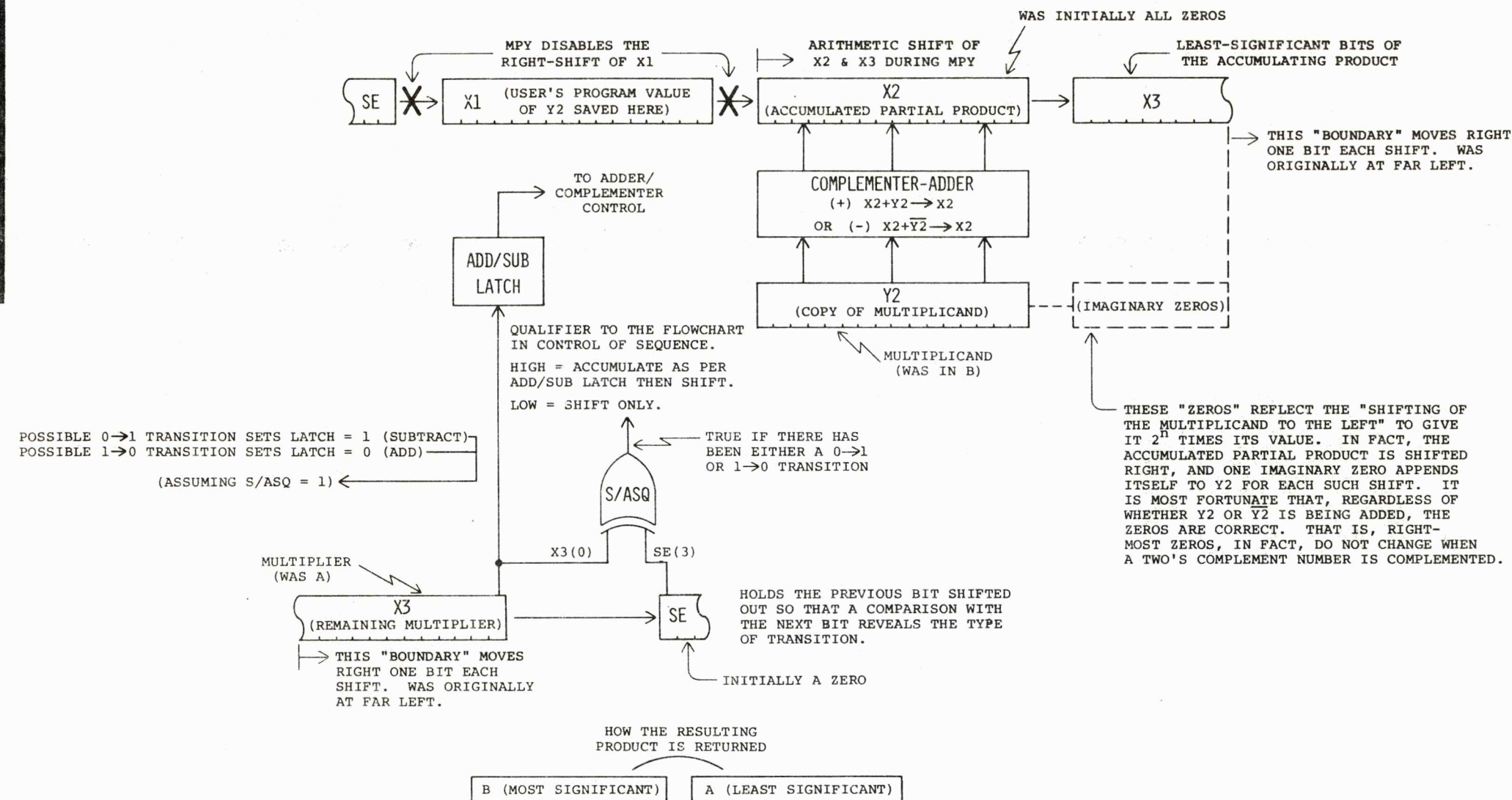


FIG 14-14-5

SECTION 14 (CONTINUED)

Figure 14-14-5 is a block diagram that shows what hardware elements are involved in MPY and how they operate together at the behest of the flow chart. The special hardware attributes called into play during MPY are as follows.

The shift connection between SE, X1 and X2 is broken. The right shift of X1 during shifts of X2 and X3 is prevented. This is done by Shift Control's separation of a right shift operation into SR (which goes to X2 and X3) and SR1 (which goes only to X1). During an MPY SR1 is disabled.

Next, the most significant bit of X2 is latched upon itself so that shifts of X2 and X3 produce an

arithmetic right shift.

The connection between DC and the carry-in line to the Adder is broken and the Add/Sub Latch replaces DC for that function. The Add/Sub Latch responds to possible zero-to-one transitions, and causes a carry-in and a one's complement of the multiplicand, which further causes the Adder to subtract. Other possible transitions result in no carry-in and no complementing, causing the Adder to add. However, additions and subtractions generated by the Adder must be explicitly used by the flow chart via a DMP ADD/SET X2, which must then be followed by a shift. Also, not all cases that cause the Adder

to add result in actual consummated additions.

The Qualifier S/ASQ (Shift/Accumulate-Shift) detects the occurrence of either a zero-to-one or one-to-zero transition and informs the flow chart whether to accumulate and shift or simply shift. A zero-to-one transition needs an accumulation by subtraction and then a shift. A one-to-zero transition needs an accumulation by addition followed by a shift. All other transitions require only a shift. S/ASQ differentiates between those cases that need accumulation (the type will be specified by the Add/Sub Latch) and those that don't. The flow chart implements

the difference between the two cases involving accumulation.

Returning now to the flow chart of Figure 14-14-1, the N-Counter equals 15 upon entry. It will be used as a loop counter for the main multiplication loop. The first thing that is done is to save the contents of Y2, so that it may receive the multiplicand from B. The multiplier in A is put into X3.

Next, the shift/accumulate-shift loop is entered. This actually performs the multiplication. After each pass through the loop the N-Counter is decremented. When the bottom of the loop is reached with the N-Counter already zero the loop is exited.

Last, the necessary registers are restored and the result written into the B and A registers.

There is a bug concerning the operation of MPY. This bug was discovered only after a long time (in November '77). It concerns the operation of the algorithm when one of the numbers is 2^{15} (a 1 followed by 15 0's). Briefly, the symptoms of the bug are these. Assuming that 2^{15} should represent -32,768, the result of multiplication by that number will have the incorrect sign if 2^{15} was in B (including when A also equals 2^{15}). If 2^{15} is in A (but not in B) then the correct sign is obtained. All of this is predicated, of course, on the assumption that 2^{15} is to represent -32,768. One could choose that it should represent +32,768. Then the results are opposite with respect to the starting location of the 2^{15} .

"Well," you might argue, "what difference does it make, since correct results depend upon an arbitrary assignment of the meaning of 2^{15} , and since the hardware can't be constructed to meet both definitions?"

True, but $B \times A$ ought to be the same as $A \times B$.

MPY BUG: $N \cdot 2^{15} = -2^{15} \cdot N$

CDC SEGMENT OF THE EMC ASM CHART

CDC

BUG IN MPY

HARDWARE NEEDED
FOR MPY

CASE I

1. LET: B = 1 000 000 000 000 000 (MULTIPLICAND=±32768)
A = 0 000 000 000 000 001 (MULTIPLIER=1)
2. SEQUENCE OF MULTIPLIER TRANSITIONS:
- 0 000 000 000 000 001 | 0
- ← FOR INITIAL COMPARISON PURPOSES
ZERO-TO-ONE
ONE-TO-ZERO
14 ZERO-TO-ZERO TRANSITIONS
3. ZERO-TO-ONE TRANSITION; SUBTRACT B FROM ACCUMULATION (i.e., 2'S COMPLEMENT B AND ADD IT TO ZERO-WHICH IS THE INITIAL VALUE OF THE ACCUMULATION). THEN SHIFT ACCUMULATION RIGHT ONCE.
- COMPLEMENT { 0 111 111 111 111 111 ← 1'S COMPLEMENT OF B
 + 1 ADD ONE
 —————
 1 000 000 000 000 000 ← 2'S COMPLEMENT OF B:

ADD { + 0 NOTE THAT THERE IS NO CHANGE-
 ————— 2'S COMPLEMENT OF ±32768 IS ±32768
 1 000 000 000 000 000

SHIFT { 1 100 000 000 000 000 | 0 --- -- -- -- --
4. ONE-TO-ZERO TRANSITION: ADD B TO ACCUMULATION, THEN SHIFT.
- THIS NEVER HAPPENS →
- ADD { 1 100 000 000 000 000 | 0 --- -- -- -- --
 + 1 000 000 000 000 000
 —————
 1 0 100 000 000 000 000 | 0 --- -- -- -- --

SHIFT { 0 010 000 000 000 000 | 0 0-- -- -- -- --
5. NOW THERE ARE 14 ZERO-TO-ZERO TRANSITIONS, EACH ACCOMPANIED BY A SHIFT.
- AFTER 14 SHIFTS { 0 000 000 000 000 000 | 1 000 000 000 000 000 | ←
- 32-BIT RESULT

FIG 14-14-6

CASE II

1. LET: B = 0 000 000 000 000 001 (MULTIPLICAND=1)
A = 1 000 000 000 000 000 (MULTIPLIER=±32768)
2. SEQUENCE OF MULTIPLIER TRANSITIONS:
- 1 000 000 000 000 000 | 0 ← FOR INITIAL COMPARISON PURPOSES
- 15 ZERO-TO-ZERO TRANSITIONS
- ZERO-TO-ONE
3. FIRST THERE ARE 15 ZERO-TO-ZERO TRANSITIONS, EACH ACCOMPANIED BY A SHIFT.
BUT THE ACCUMULATION IS ZERO TO BEGIN WITH, AND SO IT REMAINS ZERO.
- AFTER 15
SHIFTS { 0 000 000 000 000 000 | 0 000 000 000 000 00-
4. ZERO-TO-ONE TRANSITION; 2'S COMPLEMENT B AND ADD TO ACCUMULATION, THEN SHIFT.
- COMPLEMENT { 1 111 111 111 111 110 ← 1'S COMPLEMENT OF B
+ 1 ADD ONE
1 111 111 111 111 111 ← 2'S COMPLEMENT OF B
- ADD { + 0 000 000 000 000 000 | 0 000 000 000 000 00-
1 111 111 111 111 111 | 0 000 000 000 000 00-
- SHIFT { 1 111 111 111 111 111 | 1 000 000 000 000 000
- 32-BIT RESULT
5. THE RESULT ABOVE IS THE NEGATIVE OF THE RESULT IN THE OTHER CASE.
MULTIPLICATION WITH ±32768 IS NOT COMMUTATIVE.
- 0 000 000 000 000 000 | 0 111 111 111 111 111 ← 1'S COMPLEMENT OF
RESULT IN CASE II
- + 1 ADD ONE
- 0 000 000 000 000 000 | 1 000 000 000 000 000 ← 2'S COMPLEMENT OF
RESULT IN CASE II IS
SAME AS RESULT IN
CASE I

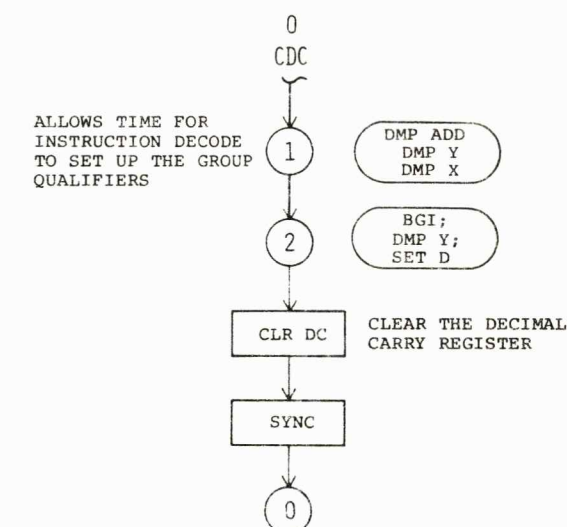


FIG 14-15

SECTION 14 (CONTINUED)

Figure 14-14-6 illustrates a simple case where $A \times B$ does not equal $B \times A$ when one of the numbers is 2^{15} . For now, no fix is contemplated. Your best bet is to understand the bug, and go around it with traps in the software for the failure cases, if they are apt to occur.

Figure 14-15 is the ASM chart segment for the CDC (Clear Decimal Carry) machine-instruction. By this point in the book it shouldn't be necessary to explain something like this.

"NON-EMC INSTRUCTION" SEGMENT OF THE EMC ASM CHART

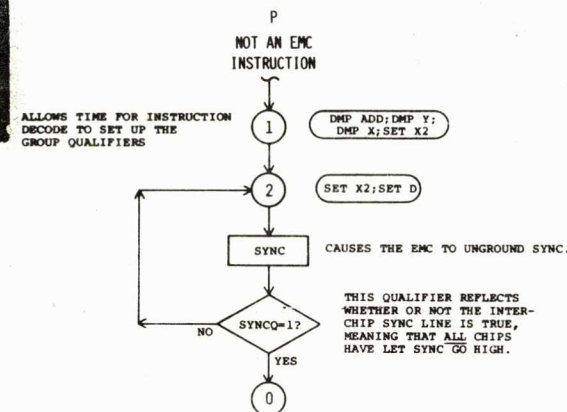


FIG 14-16

SECTION 14 (CONTINUED)

Figure 14-16 illustrates the segment of the ASM chart reached when the fetched machine-instruction is not an EMC machine-instruction. The EMC issues the SYNC micro-instruction. This, however, does not necessarily cause the inter-chip line called SYNC to go true. That event will occur when the chip for which the machine-instruction did apply is finished and also issues SYNC. The EMC idles in a loop around state two until the inter-chip SYNC line is true, and all chips are expecting the next memory cycle to be an instruction fetch. By idling in state two the EMC ignores all activity except memory cycles addressed to its own internal registers. As soon as it is known that the next memory cycle will be the instruction fetch the EMC transitions to state zero to capture and decode the instruction.

RELATIONSHIP OF M-SECTION ACTIVITY TO THE MAIN EMC STATE MACHINE, OR TO THE OUTSIDE WORLD.

CHOOSE ONE OF THE FOLLOWING POSSIBILITIES:

1. EMC ORIGINATES A READ TO ITSELF: USE I AND II
2. EMC ORIGINATES A WRITE TO ITSELF: USE II AND III
3. EXTERNAL AGENCY ORIGINATES A READ OF AN EMC REGISTER: USE II (SOMETHING IN THE EXTERNAL AGENCY CORRESPONDS TO I)
4. EXTERNAL AGENCY ORIGINATES A WRITE TO AN EMC REGISTER: USE II (SOMETHING IN THE EXTERNAL AGENCY CORRESPONDS TO III)

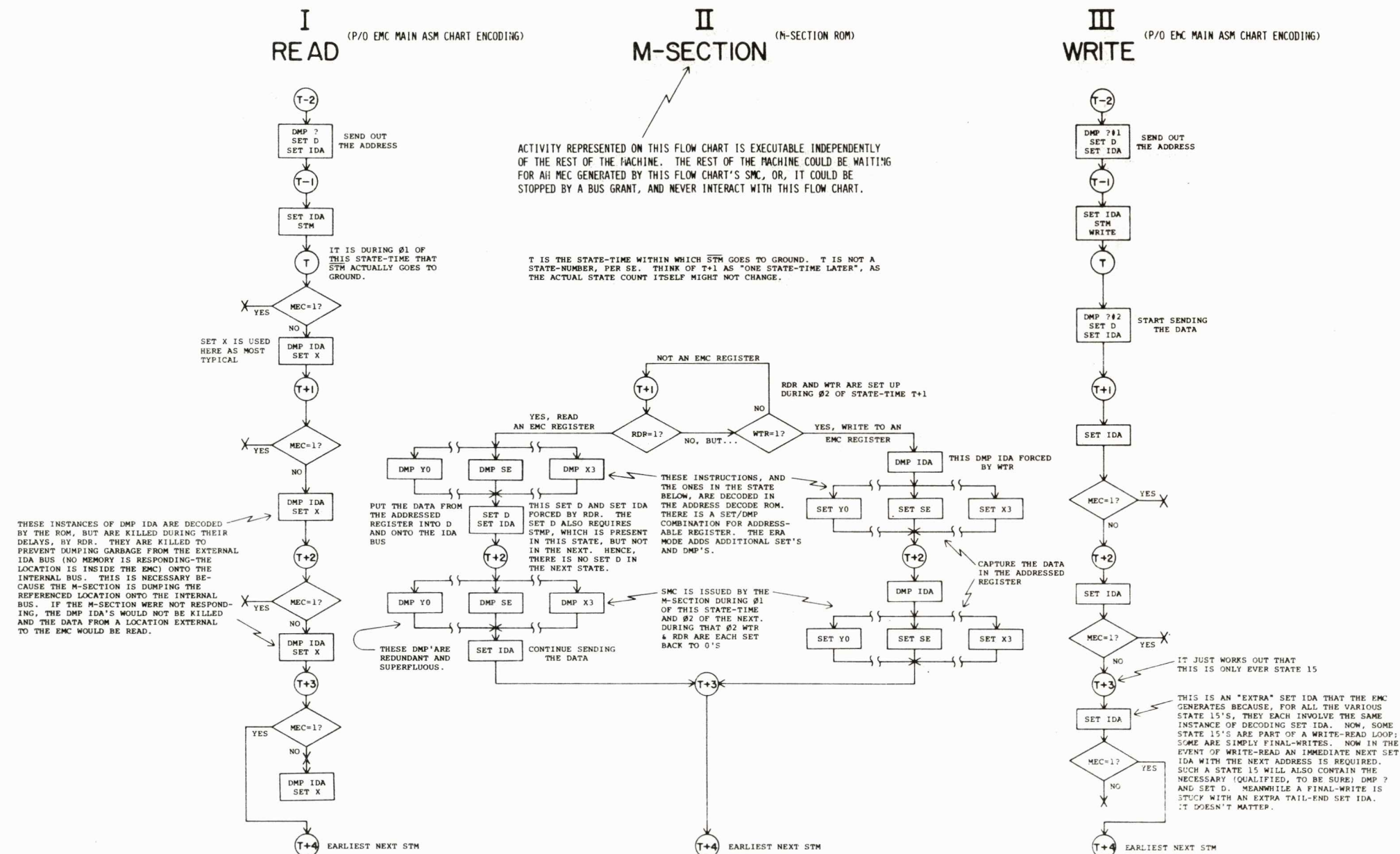


FIG 14-17

Figure 14-17 depicts the nature of the M-Section's response to a memory cycle directed to a register internal to the EMC. The flow chart is usable regardless of whether the memory cycle was originated by the EMC itself, or by an external agency.

Section II of the figure is the actual M-Section activity. This activity is fully independent of all the rest of the EMC, and its state numbers are therefore shown as relative.

CONVENTIONS USED IN THE WAVEFORMS

1.

OR

TRANSITION UP OR TRANSITION DOWN CAN OCCUR ANYTIME WITHIN THE INDICATED INTERVAL. USED TO INDICATE TIME-WINDOWS WITHIN WHICH EXTERNALLY ORIGINATED EVENTS CAN HAPPEN. REPRESENTS IDEALIZED LOGICAL ACTIVITY; RISE TIMES AND DELAYS ARE TAKEN INTO CONSIDERATION ONLY IN A GENERAL WAY.
2.

REPRESENTS THE SET-UP TIME OF A SIGNAL BEING DRIVEN.
3.

REPRESENTS A LINE THAT IS EITHER UNDEFINED OR A DON'T CARE.
4.

REPRESENTS A LINE THAT IS ACTIVELY PULLED-UP.
5.

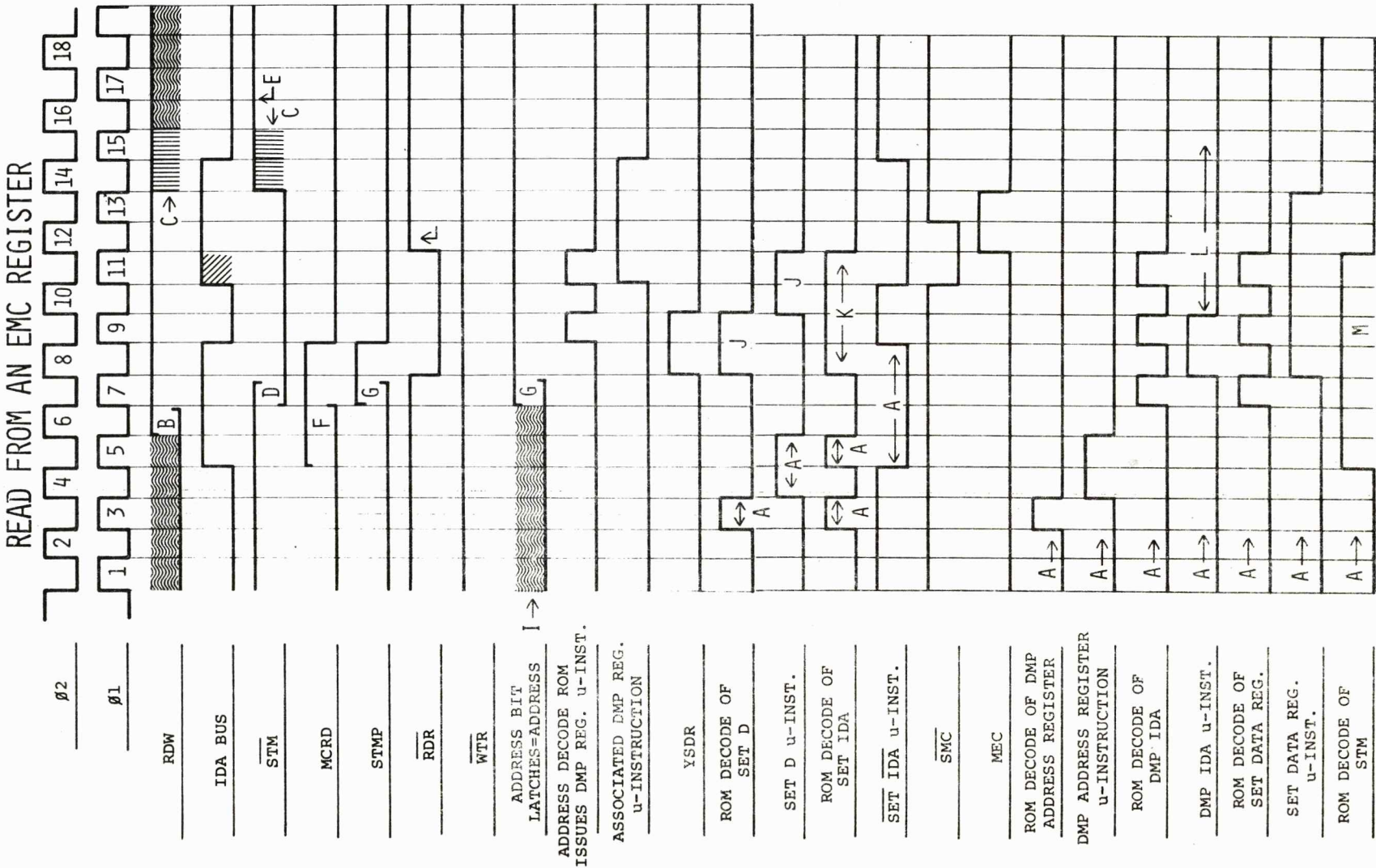
CAPITAL LETTERS FROM THE START OF THE ALPHABET REPRESENT EXPLANATORY NOTES.
6.

NUMERALS IN THE $\phi 2$ - $\phi 1$ WAVEFORMS ARE STRICTLY FOR REFERENCE WITHIN THAT PARTICULAR SET OF WAVEFORMS, AND HAVE NO SIGNIFICANCE OUTSIDE THAT SET.
7.

DOTTED LINES INDICATE ZERO OR MORE COMPLETE STATE TIMES THAT OCCUR AS A FUNCTION OF SOME EXTERNAL CONDITION (SUCH AS WAITING FOR MEMORY COMPLETE).
8.

IN GENERAL, THE WAVEFORMS ARE QUITE IDEALIZED. THEY EXPLAIN THE LOGICAL RELATIONSHIPS BETWEEN SIGNALS. BUT ACTUAL DELAYS, RISE TIMES, SIGNAL LEVELS AND THRESHOLDS ARE NOT INDICATED.

FIG 15-1



- NOTES
- A. PRESENT ONLY IF THE EMC IS THE ORIGINATOR OF THE MEMORY CYCLE.
 - B. IF THIS READ MEMORY CYCLE IMMEDIATELY FOLLOWS A PREVIOUS WRITE CYCLE, THE START OF THIS $\phi 2$ IS WHEN RDW WILL GO HIGH.
 - C. ACTIVE PULL-UP BY THE BPC, IN RESPONSE TO SMC.
 - D. TRANSITIONS AT THE START OF $\phi 1$ IF THE EMC IS THE ORIGINATOR. AN EXTERNAL AGENCY HAS UNTIL PRIOR TO $\phi 2$.
 - E. EARLIEST NEXT STM.
 - F. DEPENDS UPON WHEN THE ADDRESS ON THE IDA BUS STABILIZES.
 - G. FOLLOWS STM.
 - H. RESET BY SMC.
 - I. REFLECTS THE PREVIOUS LATCHED ADDRESS.
 - J. A RESULT OF YSDR.
 - K. CAUSED BY RDR.
 - L. RDR DISABLES DMP IDA, DURING ITS DELAY, TO PREVENT INTERFERENCE WITH THE DMP REGISTER INSTRUCTION ISSUED BY THE ADDRESS DECODE ROM
 - M. SELF-LATCHING ROM OUTPUT RESET BY MEC.

FIG 15-2

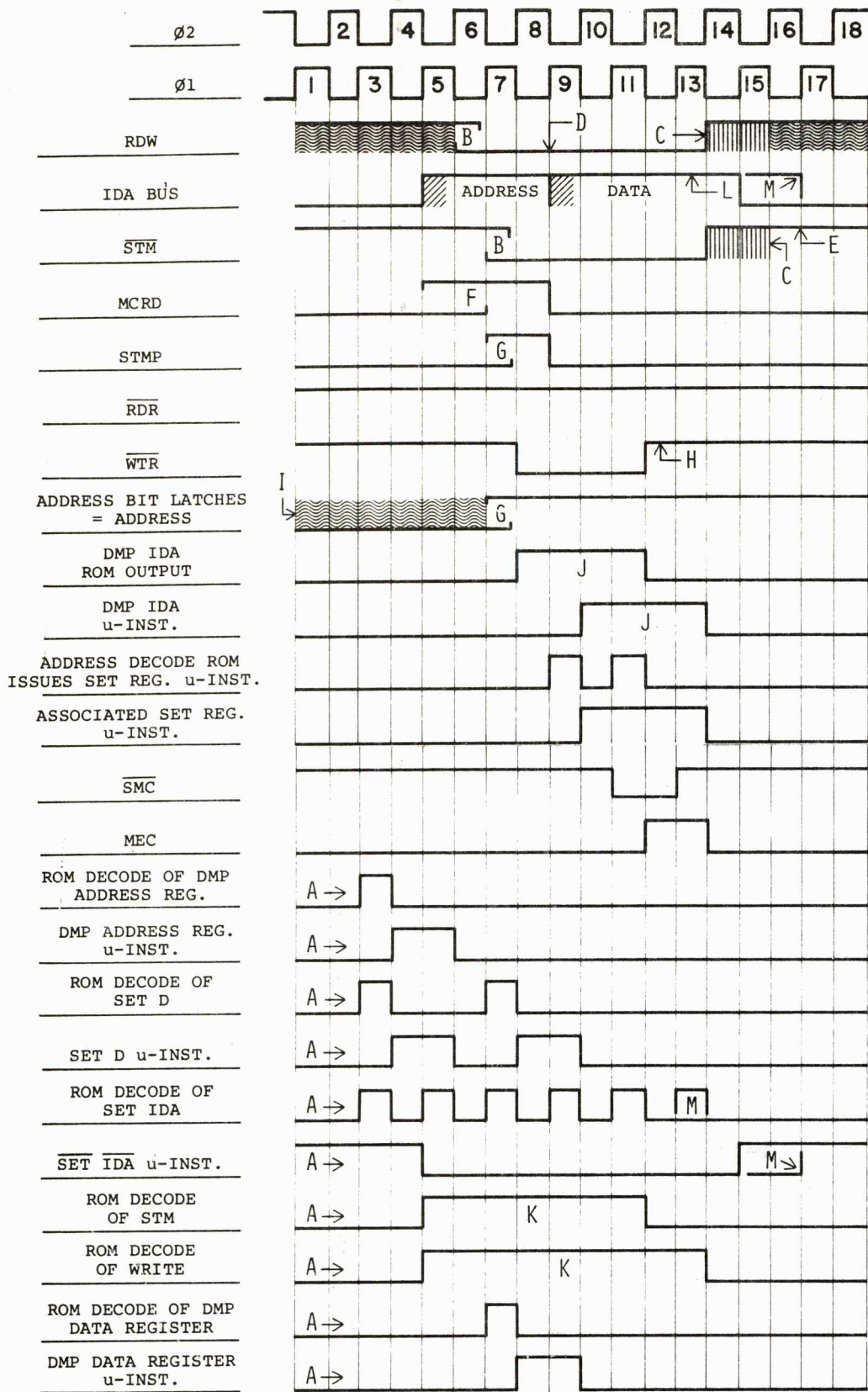
NON-EMC
INSTRUCTION

M-SECTION
FLOWCHART

WAVEFORM
CONVENTIONS

EMC REGISTER
READ

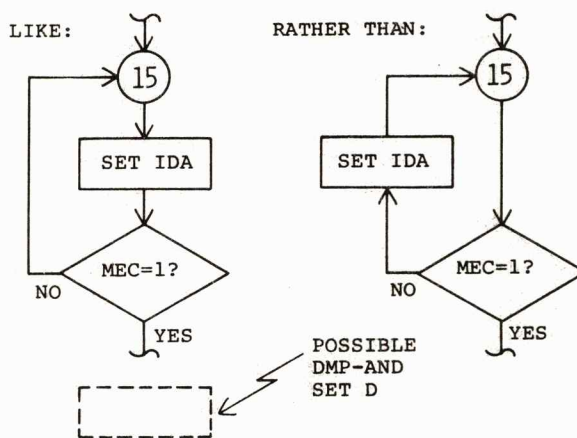
WRITE TO AN EMC REGISTER



NOTES

- A. PRESENT ONLY IF THE EMC IS THE ORIGINATOR OF THE MEMORY CYCLE.
- B. TRANSITIONS AT THE START OF ø1 IF THE EMC IS THE ORIGINATOR. AN EXTERNAL AGENCY HAS UNTIL PRIOR TO ø2.
- C. ACTIVE PULL-UP BY THE BPC, IN RESPONSE TO SMC.
- D. EARLIEST RELEASE OF RDW IF AN EXTERNAL AGENCY IS ORIGINATING THE MEMORY CYCLE.
- E. EARLIEST NEXT STM.
- F. DEPENDS UPON WHEN THE ADDRESS ON THE IDA BUS STABILIZES.
- G. FOLLOWS STM.
- H. RESET BY SMC.
- I. REFLECTS THE PREVIOUS LATCHED ADDRESS.
- J. CAUSED BY WTR.
- K. SELF-LATCHING ROM OUTPUT RESET BY MEC.
- L. WAVEFORM DEPICTED ILLUSTRATES THE EMC AS AN ORIGINATOR. IF AN EXTERNAL AGENCY WERE THE ORIGINATOR IT MUST PRESENT ROCK-SOLID DATA ALL DURING CLOCK-TIME 13, WHICH IS WHEN THE DATA IS LATCHED FOR THE FINAL TIME.

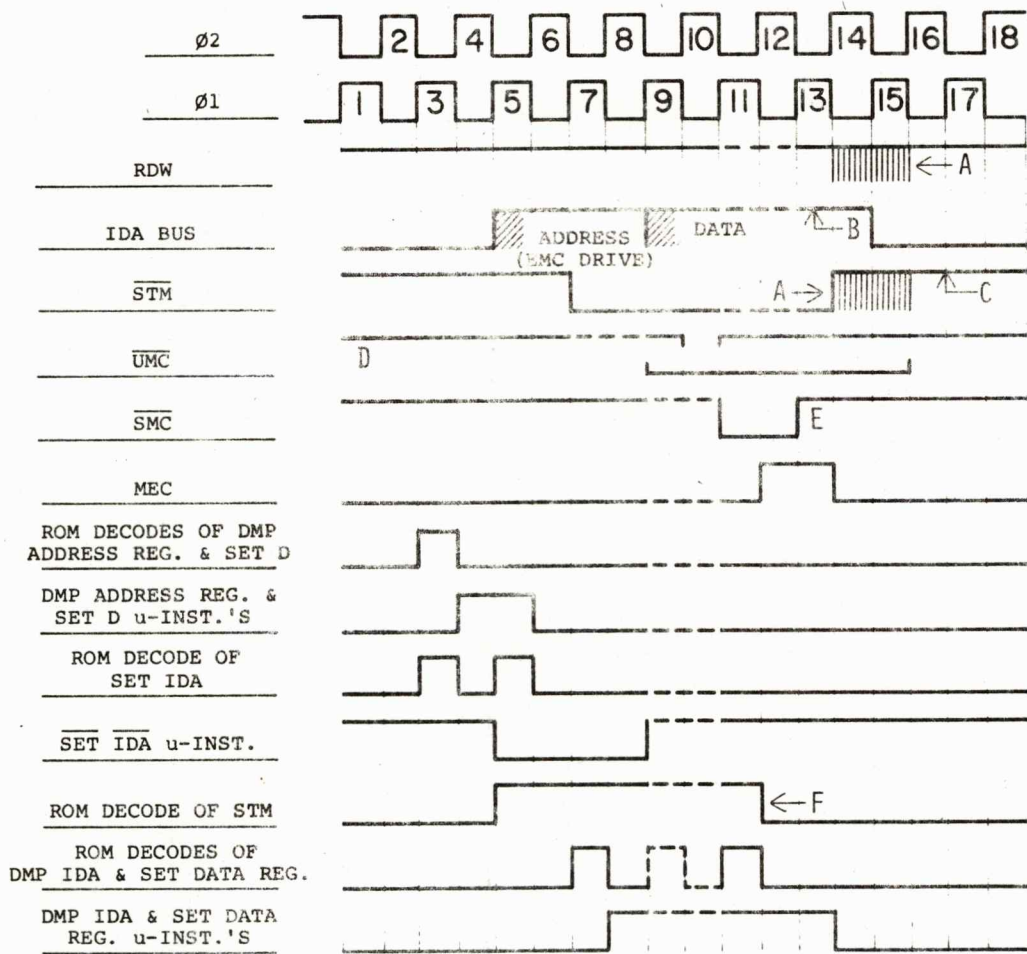
- M. IF THE EMC IS THE ORIGINATOR OF A WRITE MEMORY CYCLE, ONE EXTRA SET IDA IS GIVEN, BECAUSE THE ONLY STATES DOING SUCH MEMORY CYCLES LOOK



THE INFORMATION PLACED ON THE IDA BUS BY THIS LAST SET IDA MAY BE DIFFERENT THAN THE DATA SENT AS PART OF THE MEMORY CYCLE. FOR INSTANCE, IT MAY BE AN ADDRESS FOR ANOTHER MEMORY CYCLE.

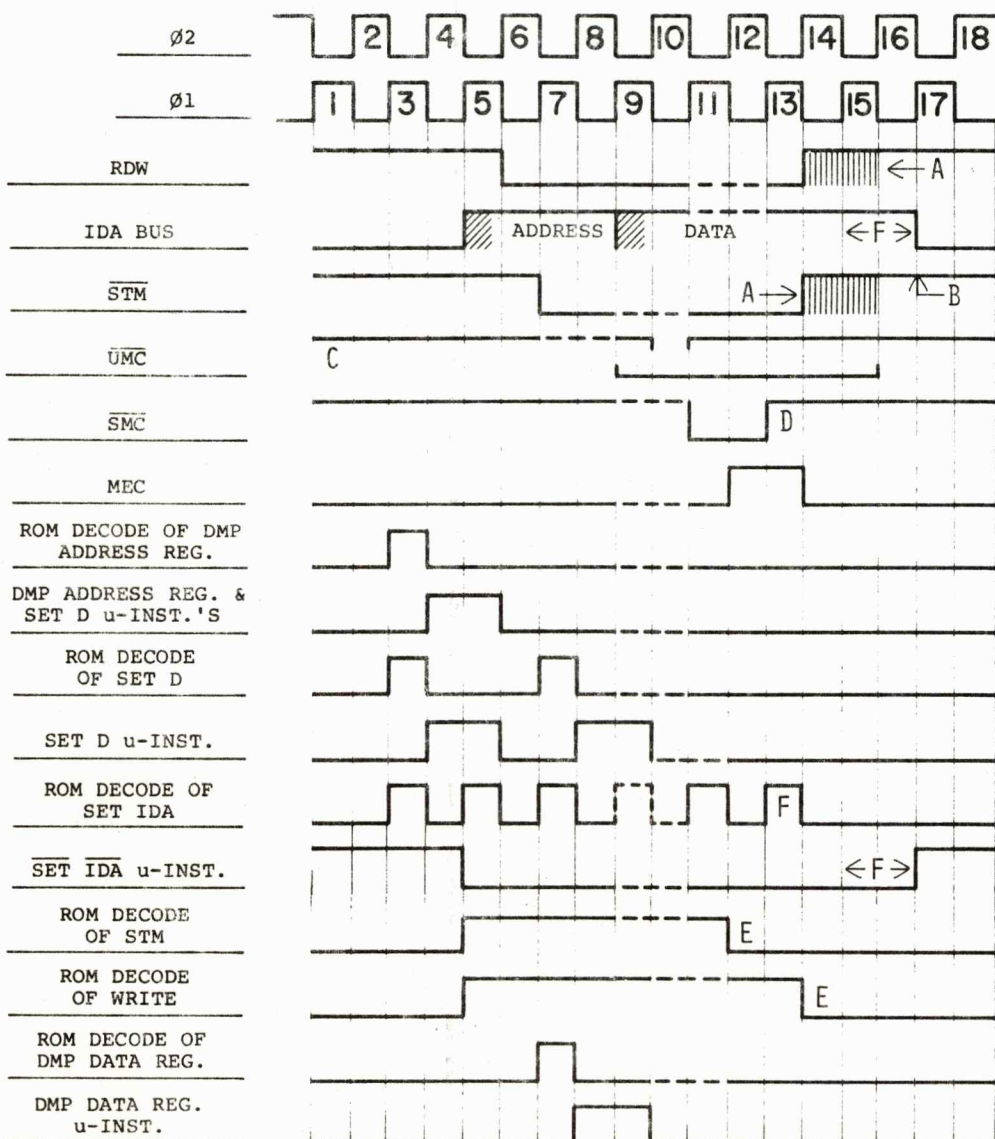
FIG 15-3

FIG 15-4



- A. ACTIVE PULL-UP BY THE BPC, IN RESPONSE TO SMC.
- B. DATA IS LATCHED FOR THE FINAL TIME DURING THIS CLOCK-TIME.
- C. EARLIEST NEXT STM.
- D. USE OF UMC IS OPTIONAL. MEMORY MAY ISSUE SMC EXACTLY AS SHOWN, IF DESIRED.
- E. SMC IS GENERATED EITHER BY THE BPC (AS A SERVICE FUNCTION IN RESPONSE TO UMC), OR DIRECTLY BY THE MEMORY ITSELF.
- F. SELF-LATCHING ROM OUTPUT RESET BY MEC.

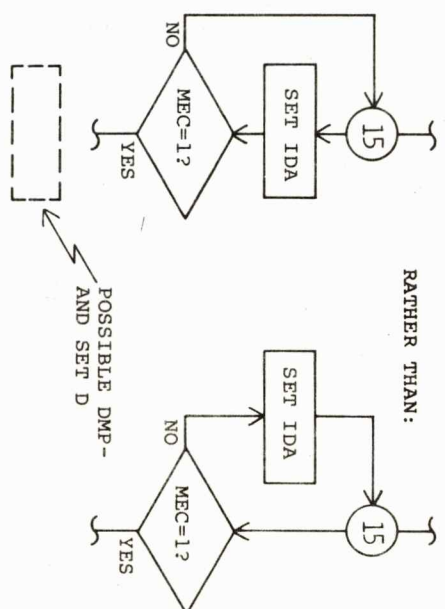
FIG 15-5



- A. ACTIVE PULL-UP BY THE BPC, IN RESPONSE TO SMC.
- B. EARLIEST NEXT STM.
- C. USE OF UMC IS OPTIONAL. MEMORY MAY ISSUE SMC EXACTLY AS SHOWN, IF DESIRED.
- D. SMC IS GENERATED EITHER BY THE BPC (AS A SERVICE FUNCTION IN RESPONSE TO UMC), OR DIRECTLY BY THE MEMORY ITSELF.
- E. SELF-LATCHING ROM OUTPUT RESET BY MDC.

IF THE EMC IS THE ORIGINATOR OF A WRITE MEMORY CYCLE, ONE EXTRA SET IDA IS GIVEN, BECAUSE THE ONLY STATES DOING SUCH MEMORY CYCLES LOOK LIKE: T

RATHER THAN: T



THE INFORMATION PLACED ON THE IDA BUS BY THIS LAST SET IDA MAY BE DIFFERENT THAN THE DATA SENT AS PART OF THE MEMORY CYCLE. FOR INSTANCE, IT MAY BE AN ADDRESS FOR ANOTHER MEMORY CYCLE.

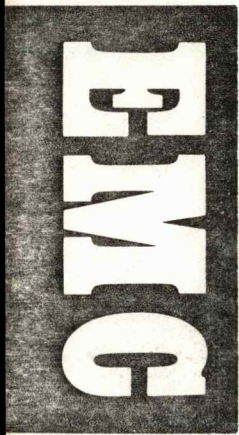


TABLE OF EMC MAIN CONTROL ROM CONTENTS

INPUT QUALIFIERS TO THE ROM

MICRO INSTRUCTION	GROUP					STATE				OTHER QUALIFIERS	INSTRUCTION IS ISSUED IN ANY COMBINATION OF:	
	IL4Q	IL3Q	IL2Q	IL1Q	ILOQ	S3Q	S2Q	S1Q	S0Q		GROUPS	STATES
CLR AE	-	-	-	-	-	0	0	0	0		A-P	0
CLR DC	-	0	-	-	-	1	1	0	0	STP	H-M	12
CLR DC	-	1	-	1	-	-	1	1	-		E-G,0,P	5,5,9,10
CLR DC	1	1	0	-	-	0	-	1	-		C-F	5,6,13,14
CLR DC	0	-	1	1	-	-	-	1	0		0	2,6,10,14
CLR DC	-	0	-	1	-	-	0	0	1	MEC·WP1Q	J	3,15
CLR DC	0	-	-	0	1	-	-	0	1	STP·WP1Q	K	3,7,11,15
DEC WP	0	0	-	-	-	1	1	0	-	STP	H-K	11,12
DEC WP	-	1	0	-	0	1	0	1	0	STP	B,C,F,G	2
DEC WP	-	1	0	-	-	0	0	1	1	STP	B-G	13
DEC WP	-	0	-	1	-	1	-	-	0	STP·WP2Q	J	2,4,10,12
DEC WP	-	1	-	1	-	0	-	1	1	STP·WP2Q	E-G,0,P	5,13
DEC WP	0	-	-	1	-	-	0	0	1	STP·MEC·WP1Q	G,J,0	3,15
DEC WP	1	-	0	-	-	0	-	0	1	STP·MEC·WP1Q	C-F,L,M	7,15
DEC WP	-	0	-	-	-	1	1	0	1	STP·WP1Q	H-M	11
DEC WP	1	-	1	-	-	1	0	0	0	STP·MEC·WP3Q	N,P	4
DEC N	1	-	-	-	-	0	-	0	1	STP·MEC·WP1Q	C-F,L-P	7,15
DEC N	-	1	0	-	-	-	1	0	0	STP·WPOQ	B-G	8,12
DEC N	-	-	-	-	-	0	0	0	1	STP·MEC	A-P	15
DEC N	-	-	-	1	-	-	1	1	0	STP	E-G,J,0,P	6,10
DEC N	-	-	-	-	-	1	1	0	0	STP	A-P	12
DMP ADD	0	1	-	-	-	-	0	-	1	STP·MEC	A,B,G,0	1,3,13,15
DMP ADD	-	0	-	0	-	1	1	-	1	STP	H,I,K,L,M	9,11
DMP ADD	-	0	-	1	-	0	-	1	0	STP	J	6,14
DMP ADD	-	-	-	-	-	-	0	1	1	STP	A-P	1,13
DMP ADD	-	-	1	-	-	-	1	1	-	STP	A,H,N-P	5,6,9,10
DMP AE	1	0	-	-	-	0	0	1	-	STP	L,M	13,14
DMP AE	0	1	-	1	-	0	0	1	-	STP	G,0	13,14
DMP ADR1	-	0	1	-	-	-	-	1	-	STP	H	1,2,5,6,9,10,13,14

FIG 16-1

TABLE OF EMC MAIN CONTROL ROM CONTENTS (CONT.)

INPUT QUALIFIERS TO THE ROM

MICRO INSTRUCTION	GROUP					STATE				OTHER QUALIFIERS	INSTRUCTION IS ISSUED IN ANY COMBINATION OF:	
	IL4Q	IL3Q	IL2Q	IL1Q	IL0Q	S3Q	S2Q	S1Q	S0Q		GROUPS	STATES
DMP ADR1	-	0	-	1	-	0	1	1	-	STP	J	5,6,
DMP ADR1	1	1	-	0	-	0	1	1	-	STP	C,D,N	5,6
DMP ADR1	1	-	-	-	-	0	1	-	0	STP	C-F,L-N,P	6,8
DMP ADR1	1	0	-	-	-	-	1	1	-	STP·NZQ	L,M	5,6,9,10
DMP ADR1	-	1	0	-	-	1	-	0	1	STP·MEC·WPOQ	B-G	3,11
DMP IDA	-	0	-	-	-	0	1	0	1	STP·MEC	H-M	7
DMP IDA	-	-	-	0	-	0	1	0	1	STP·MEC	A-D,H,I,K-N	7
DMP IDA	-	-	-	-	-	-	0	0	0	STP·MEC	A-P	0,4
DMP ONE	1	-	1	-	-	1	0	0	0	STP·MEC	N,P	4
DMP ONE	-	-	0	0	0	1	0	1	-	STP	B,C,I,L	1,2
DMP ONE	1	0	-	-	1	1	-	-	0	STP	M	2,4,10,12
DMP ONE	1	-	0	-	0	1	0	1	-	STP	C,F,L	2
DMP ONE	0	-	-	1	-	1	-	0	-	STP	G,J,0	3,4,11,12
DMP SE	1	1	0	-	-	0	0	1	-	STP	C-F	13,14
DMP X	-	-	-	1	-	-	0	-	1	STP	E-G,J,0,P	1,3,13,15
DMP X	-	-	0	-	-	1	1	1	-	STP	B-G,I-M	9,10
DMP X	1	-	1	-	-	0	-	1	-	STP	N,P	5,6,13,14
DMP X	-	-	-	1	0	-	-	1	0	STP·D1ZQ	F,G,0	3,7,11,15
DMP X1	0	-	0	0	-	0	-	1	0	STP	B,I,K	6,14
DMP X1	-	1	1	-	-	-	1	0	-	STP	A,N,0,P	7,8,11,12
DMP X2	0	1	-	-	-	-	1	0	-	STP·MEC	A,B	7,8,11,12
DMP X3	0	-	-	-	-	-	1	1	-	STP	A,B,H,I,K	5,6,9,10
DMP Y	0	-	-	-	-	1	0	-	-	STP	G,J,0	1-4
DMP Y	-	1	-	1	-	0	-	1	1	STP	E-G,0,P	5,13
DMP Y	1	-	1	-	-	1	-	-	1	STP	N,P	1,3,9,11
INC AE	-	0	-	-	1	1	-	0	0	STP·COUTQ	J,K,M	4,12
INC AE	0	-	-	-	-	-	-	0	1	STP·D1ZQ·AE12Q	A,B,G-K,0	3,7,11,15
INC AE	-	0	-	-	0	1	1	0	0	STP·COUTQ	H,I,L	12
INC WP	-	1	0	-	-	0	0	0	1	STP·MEC	B-G	15

FIG 16-2

TABLE OF EMC MAIN CONTROL ROM CONTENTS (CONT.)

INPUT QUALIFIERS TO THE ROM

MICRO INSTRUCTION	GROUP					STATE				OTHER QUALIFIERS	INSTRUCTION IS ISSUED IN ANY COMBINATION OF:	
	IL4Q	IL3Q	IL2Q	IL1Q	ILOQ	S3Q	S2Q	S1Q	S0Q		GROUPS	STATES
INC WP	-	-	-	1	-	1	1	-	=	STP	E-G, J, O, P	9-12
INC WP	0	-	-	1	0	-	-	0	1	STP	G, O	3, 7, 11, 15
INC WP	-	1	0	-	-	1	1	-	0	STP	B-G	10, 12
INC WP	1	-	-	1	-	-	1	1	0	STP	E, F, P	6, 10
INC WP	1	1	0	-	-	-	1	0	0	STP · NZQ	C-F	8, 12
INC WP	1	-	1	-	-	0	-	1	0	STP	N, P	6, 14
INC WP	-	1	0	-	-	1	0	0	0	STP · MEC · WP3Q	B-G	4
INC WP	-	1	0	-	-	-	1	0	0	STP · WPOQ	B-G	8, 12
INC WP	0	-	-	1	-	-	0	0	1	STP · MEC · WP1Q	G, J, O	3, 15
INC WP	1	1	0	-	-	0	-	0	1	STP · MEC · WP1Q	C-F	7, 15
NS0	-	-	-	-	-	0	0	0	0	MEC	A-P	0
NS0	0	-	1	1	-	-	-	1	0		0	2, 6, 10, 14
NS0	-	-	1	1	-	-	-	1	0	SYNCQ	O, P	2, 6, 10, 14
NS0	0	0	-	1	-	1	1	-	-	WP3Q	J	9-12
NS0	0	0	-	-	-	1	1	0	-	WP1Q	H-K	11, 12
NS2	1	-	1	1	-	-	-	1	-	SYNCQ	P	1, 2, 5, 6, 9, 10, 13, 14
NS2	-	0	-	1	0	1	0	-	-	WP2Q	J	1-4
NS2	-	0	1	-	-	-	1	0	-	WP1Q	H	7, 8, 11, 12
NS3	-	1	0	-	-	1	0	0	0	MEC · WP3Q	B-G	4
NS3	1	-	1	-	0	1	0	-	0	MEC · WP3Q	N	2, 4
NS4	-	-	-	-	-	1	0	0	0	MEC	A-P	4
NS5	-	0	-	1	-	-	0	0	1	MEC · WP1Q	J	3, 15
NS5	-	0	-	1	-	1	-	-	0	WP2Q	J	2, 4, 10, 12
NS5	1	-	-	0	1	1	-	-	0		D, M	2, 4
NS5	0	-	0	1	0	1	1	1	0		G	2, 10
NS5	0	-	0	0	-	0	0	0	1	MEC · NZQ	B, I, K	15
NS5	-	1	-	1	-	0	1	1	1	WP2Q	E-G, O, P	5
NS5	1	-	-	-	-	0	0	0	1	MEC · WP1Q	C-F, L-N, P	7
NS7	-	0	-	-	-	0	0	0	1	MEC	H-M	7

FIG 16-3

TABLE OF EMC MAIN CONTROL ROM CONTENTS (CONT.)

INPUT QUALIFIERS TO THE ROM

MICRO INSTRUCTION	GROUP					STATE				OTHER QUALIFIERS	INSTRUCTION IS ISSUED IN ANY COMBINATION OF:	
	IL4Q	IL3Q	IL2Q	IL1Q	ILOQ	S3Q	S2Q	S1Q	S0Q		GROUPS	STATES
NS7	-	-	-	0	-	0	0	0	1	MEC	A-D, H, I, K-N	7
NS7	0	-	-	1	0	0	-	-	0	WPOQ	G, O	6, 8
NS8	0	-	-	0	1	1	-	-	0		K	2, 4
NS8	-	0	-	-	1	1	0	0	0	COUTQ	J, K, M	4, 12
NS8	1	-	-	1	-	-	1	1	0		E, F, P	6, 10
NS8	1	0	-	-	-	0	1	1	1	NZQ	L, M	5
NS8	-	0	-	-	0	1	0	0	0	NZQ	H, I, L	12
NS8	1	1	0	-	-	-	0	0	0	NZQ	C-F	8, 12
NS8	-	0	0	-	-	1	0	0	1	WP1Q	I-M	11
NS8	-	1	0	-	-	0	0	0	0	WPOQ	B-G	8
NS9	0	0	-	-	-	1	0	0	0	MEC	H-K	4, 12
NS9	-	-	-	1	0	-	0	0	1	D1ZQ	F, G, O	7, 11
NS9	-	-	1	-	-	-	0	0	0	NZQ	A, H, N-P	8, 12
NS9	1	-	1	-	-	1	0	0	0	MEC·WP3Q	N, P	4
NS9	-	0	-	1	-	1	-	-	-	WP3Q	J	9-12
NS9	-	1	-	1	-	1	1	1	-	WP3Q	E-G, O, P	9, 10
NS9	1	1	0	-	-	1	0	0	-	MEC·WPOQ	C-F	11, 12
NS9	-	0	-	1	-	-	0	0	1	MEC·WP1Q	J	3, 15
NS11	1	1	0	-	-	1	1	0	-	MEC	C-F	11, 12
NS12	-	1	-	1	-	-	1	0	1	D1ZQ·AE12Q	E-G, O, P	7, 11
NS12	-	1	1	-	-	-	1	-	1	S/ASQ	A, N-P	5, 7, 9, 11
NS12	0	1	-	1	-	1	1	1	-	WP3Q	G, O	9, 10
NS13	0	1	-	0	-	-	1	0	-	MEC	A, B	7, 8, 11, 12
NS13	0	1	1	-	-	1	-	0	0	MEC	A, O	4, 12
NS13	-	0	-	1	-	-	1	0	0		J	8, 12
NS13	0	-	1	-	-	0	-	0	1	MEC·NZQ	A, H, O	7, 15
NS13	1	-	1	-	-	0	-	0	1	MEC·WP3Q	N, P	7, 15
NS13	1	-	-	1	-	1	1	-	-	WP3Q	E, F, P	9-12
NS13	-	1	0	-	-	1	1	0	-	MEC·WPOQ	B-G	11, 12

FIG 16-4

EMC ROM
CONTENTS (CONT.)EMC ROM
CONTENTS (CONT.)EMC ROM
CONTENTS (CONT.)EMC ROM
CONTENTS

TABLE OF EMC MAIN CONTROL ROM CONTENTS (CONT.)

INPUT QUALIFIERS TO THE ROM

MICRO INSTRUCTION	GROUP					STATE				OTHER QUALIFIERS	INSTRUCTION IS ISSUED IN ANY COMBINATION OF:	
	IL4Q	IL3Q	IL2Q	IL1Q	IL0Q	S3Q	S2Q	S1Q	S0Q		GROUPS	STATES
NS13	1	1	0	-	-	1	-	0	0	MEC·WP3Q·NZQ	C-F	4,12
NS15	-	-	-	-	-	0	0	0	1	MEC	A-P	15
SET D	0	1	-	0	-	-	1	0	-	STP·MEC	A,B	7,8,11,12
SET D	-	-	1	-	-	0	-	0	1	STP·MEC	A,H,N-P	7,15
SET D	-	1	0	-	-	1	1	0	-	STP·MEC	B-G	11,12
SET D	1	1	-	-	-	1	0	-	0	STP·MEC	C-F,N,P	2,4
SET D	-	1	0	-	-	1	-	-	0	STP·MEC	B-G	2,4,10,12
SET D	0	1	1	-	-	1	-	-	0	STP·MEC	A,0	2,4,10,12
SET D	-	-	0	0	-	-	1	0	0	STP	B-D,I,K,L,M	8,12
SET D	-	-	0	-	-	1	1	-	0	STP	B-G,I-M	10,12
SET D	-	0	-	-	-	-	1	1	1	STP	H-M	5,9
SET D	-	-	-	-	-	-	0	1	0	STP	A-P	2,14
SET D	-	-	-	0	-	0	1	1	1	STP	A-D,H,I,K-N	5
SET DC	0	-	-	-	1	1	0	-	0		J,K	2,4
SET DC	-	-	-	1	0	0	1	-	1	D1ZQ·AE12Q	F,G,0	5,7
SET IL	-	-	-	-	-	0	0	0	0	MEC	A-P	0
SET IDA	0	1	-	-	-	-	1	-	-	SSTP·MEC	A,B,G,0	5-12
SET IDA	-	1	-	-	-	1	-	0	-	SSTP·MEC	A-G,N-P	3,4,11,12
SET IDA	-	-	-	-	-	-	1	-	0	SSTP	A-P	6,8,10,12
SET IDA	-	-	0	-	-	-	1	1	-	SSTP	B-G,I-M	5,6,9,10
SET IDA	-	-	-	-	-	0	0	-	1	SSTP	A-P	13,15
SET IDA	-	-	0	-	-	-	-	1	0	SSTP	B-G,I-M	2,6,10,14
SET IDA	-	-	-	0	-	-	-	1	0	SSTP	A-D,H,I,K-N	2,6,10,14
SET IDA	-	-	-	-	-	1	-	0	1	SSTP	A-P	3,11
SET N	1	0	-	-	-	-	0	0	0	MEC	L,M	0,4
SET N	-	-	-	-	-	0	0	0	0	MEC	A-P	0
SET N	1	1	0	-	-	1	-	0	0	MEC·WP3Q	C-F	4,12
SET SE	0	1	-	1	-	-	1	1	0		G,0	6,10
SET SE	1	1	-	0	-	1	0	1	0		C,D,N	2

FIG 16-5

TABLE OF EMC MAIN CONTROL ROM CONTENTS (CONT.)

INPUT QUALIFIERS TO THE ROM

MICRO INSTRUCTION	GROUP					STATE				OTHER QUALIFIERS	INSTRUCTION IS ISSUED IN ANY COMBINATION OF:	
	IL4Q	IL3Q	IL2Q	IL1Q	IL0Q	S3Q	S2Q	S1Q	S0Q		GROUPS	STATES
SET SE	1	1	0	-	-	1	-	0	0	MEC·WP3Q	C-F	4,12
SET X	0	-	-	0	-	1	-	0	0	MEC	A,B,H,I,K	4,12
SET X	1	-	-	-	-	0	1	0	1	MEC	C-F,L-N,P	7
SET X	0	1	1	-	-	-	-	0	1	STP·MEC	A,0	3,7,11,15
SET X	0	-	0	1	-	1	0	-	0		G,J	2,4
SET X	-	1	-	1	-	0	1	1	1		E-G,0,P	5
SET X	1	-	1	-	-	-	1	1	0	STP	N,P	6,10
SET X	-	-	1	-	-	1	0	0	0	MEC·WP3Q	A,H,N-P	4
SET X1	0	1	-	0	-	-	1	0	-	MEC	A,B	7,8,11,12
SET X1	-	-	-	-	-	1	0	0	1	STP	A-P	3
SET X2	-	0	0	-	-	1	0	0	0	MEC	I-M	4
SET X2	0	1	0	-	-	0	0	0	1	STP·MEC	B,G	15
SET X2	1	-	1	-	-	1	0	1	-		N,P	1,2
SET X3	1	0	-	-	-	1	0	0	1		L,M	3
SET X3	0	-	0	0	-	0	0	1	1	STP	B,I,K	13
SET WP	-	-	-	-	-	0	0	0	0		A-P	0
SET WP	1	0	-	-	-	-	-	0	1	STP·MEC·WP1Q	L,M	3,7,11,15
SET WP	-	0	-	1	-	1	0	-	-	WP2Q	J	1-4
SET WP	1	0	-	-	-	1	-	0	1	STP·WP1Q	L,M	3,11
SET Y	-	0	-	1	-	-	1	0	1	MEC	J	7,11
SET Y	-	0	-	0	-	1	1	0	1	STP	H,I,K-M	11
SET Y	-	-	0	1	-	1	1	1	-		E,F,G,J	9,10
SET Y	1	-	1	-	-	1	0	0	0	MEC·WP3Q	N,P	4
SET Y	-	0	-	1	0	-	0	0	1	MEC·WP1Q	J	3,15
SET Y	-	0	-	1	0	-	0	0	1		J	3,15
SET Y	-	-	1	-	-	-	1	0	0	NZQ	A,H,N,0,P	8,12
SET Y	-	-	-	1	0	-	1	0	1	D1ZQ	F,G,0	7,11
SIZ	1	-	-	-	-	-	-	0	0	WPOQ	C-F,L-N,P	0,4,8,12
SRE	1	-	-	1	-	-	1	1	0	STP	E,F,P	6,10

FIG 16-6

TABLE OF EMC MAIN CONTROL ROM CONTENTS (CONT.)

INPUT QUALIFIERS TO THE ROM

MICRO INSTRUCTION	GROUP					STATE				OTHER QUALIFIERS	INSTRUCTION IS ISSUED IN ANY COMBINATION OF:	
	IL4Q	IL3Q	IL2Q	IL1Q	IL0Q	S3Q	S2Q	S1Q	S0Q		GROUPS	STATES
SRE	1	-	1	-	-	-	1	0	1	STP	N,P	7,11
SRE	-	1	0	-	-	0	1	0	0	STP·WPOQ	B-G	8
SRE	1	1	-	-	-	0	1	0	1	STP·MEC·WP1Q	C-F,N,P	7
SRE	1	1	0	-	-	-	1	0	0	STP·NZQ	C-F	8,12
SRE	-	1	1	-	-	1	1	-	1	STP·S/ASQ	A,N,O,P	9,11
SRE	-	-	-	1	0	-	1	0	1	STP·D1ZQ·AE12Q	F,G,0	7,11
STM	-	0	-	-	-	0	-	1	0	STP	H-M	6,14
STM	-	1	0	0	-	1	1	1	-	STP	B-D	9,10
STM	-	-	-	0	-	0	-	1	0	STP	A-D,H,I,K-N	6,14
STM	-	-	-	-	-	1	0	0	1	STP	A-P	3
STM	-	-	-	-	-	0	0	1	-	STP	A-P	13,14
SYNC	-	1	-	1	-	0	0	0	1	MEC	E,F,G,0,P	15
SYNC	1	-	0	-	-	0	0	0	1	MEC	C-F,L,M	15
SYNC	-	-	1	1	-	-	-	1	0		0,P	2,6,10,14
SYNC	0	0	-	-	-	1	1	0	-	WP1Q	H-K	11,12
SYNC	1	-	1	-	-	0	-	0	1	MEC·WP3Q	N,P	7,15
SYNC	0	1	-	-	-	0	0	0	1	MEC·NZQ	A,B,G,0	15
^F SYNC	-	0	-	1	-	1	1	-	-	WP3Q	J	9-12
^S SYNC	-	0	-	1	-	0	0	0	1	MEC·WP1Q	J	15
UPD DC	1	0	-	-	-	1	1	0	1	STP	L,M	11
UPD DC	-	0	-	-	0	1	1	0	1	STP	H,I,L	11
UPD DC	0	0	-	-	-	0	0	1	0	STP	H-K	14
UPD DC	-	0	-	-	-	1	1	0	1	STP·WP1Q	H-M	11
WRITE	-	1	0	0	-	1	1	1	-	STP	B-D	9,10
WRITE	-	-	-	-	-	0	0	1	-	STP	A-P	13,14
BGI	0	-	-	1	-	1	0	-	0	STP·WP2Q	G,J,0	2,4
BGI	-	1	-	1	-	0	1	1	1	STP·WP2Q	E-G,0,P	5
BGI	-	1	0	-	-	0	1	0	0	STP·WP2Q	B-G	8
^S BGI	1	0	-	-	-	1	1	0	1	STP	L,M	11

FIG 16-7

TABLE OF EMC MAIN CONTROL ROM CONTENTS (CONT.)

NOTES

- THE TABLE SHOWS WHAT "AND" CONDITIONS MUST BE MET FOR EACH INSTANCE OF DECODING THE VARIOUS MICRO-INSTRUCTIONS. FOR EACH, THE FOLLOWING GENERAL "AND" CONDITION MUST BE MET:
(SOME GROUP CONDITION) · (SOME STATE CONDITION) · (SOME QUALIFIER CONDITION)
EACH CONDITION CAN BE THE RESULT OF AN "OR" AMONG ITS CONSTITUENT ELEMENTS. THESE ARE INDICATED BY DASHES IN THE LISTING OF THE GROUP AND STATE QUALIFIER REQUIREMENTS.
- ^F DENOTES THAT THE GIVEN INSTANCE OF DECODING THE INSTRUCTION IS UNIQUE TO THE FIFTEEN BIT VERSION OF THE EMC. DELETE THAT LINE WHEN CONSIDERING THE SIXTEEN BIT VERSION.
- ^S DENOTES THAT THE GIVEN INSTANCE OF DECODING THE INSTRUCTION IS UNIQUE TO THE SIXTEEN BIT VERSION OF THE EMC. DELETE THAT LINE WHEN CONSIDERING THE FIFTEEN BIT VERSION.
- ONE'S IN THE GROUP AND STATE QUALIFIER COLUMNS, AND REQUIRED TRUE QUALIFIERS ARE ELECTRICALLY IMPLEMENTED IN THE ROM BY CONNECTING THE GATES OF DECODING TRANSISTORS ON THE VARIOUS HORIZONTAL OUTPUT LINES TO A VERTICAL DRIVE LINE THAT REPRESENTS THE NOT OF THAT QUALIFIER. ZEROS AND REQUIRED FALSE QUALIFIERS ARE IMPLEMENTED BY CONNECTING THE GATES OF THE DECODING TRANSISTORS TO THE TRUE SENSE OF THE QUALIFIER. A DASH OR UNMENTIONED QUALIFIER DENOTES THE TOTAL ABSENSE OF A DECODING TRANSISTOR.

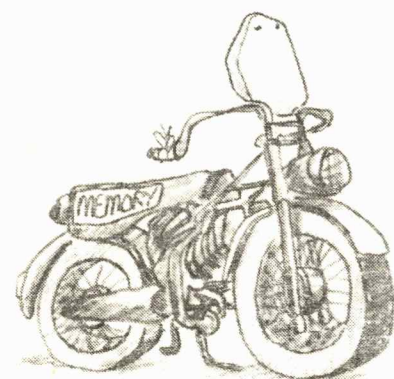
FIG 16-8

EMC ROM CONTENTS (CONT.)

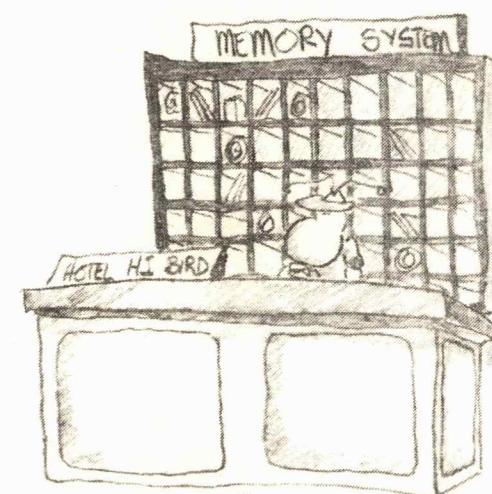
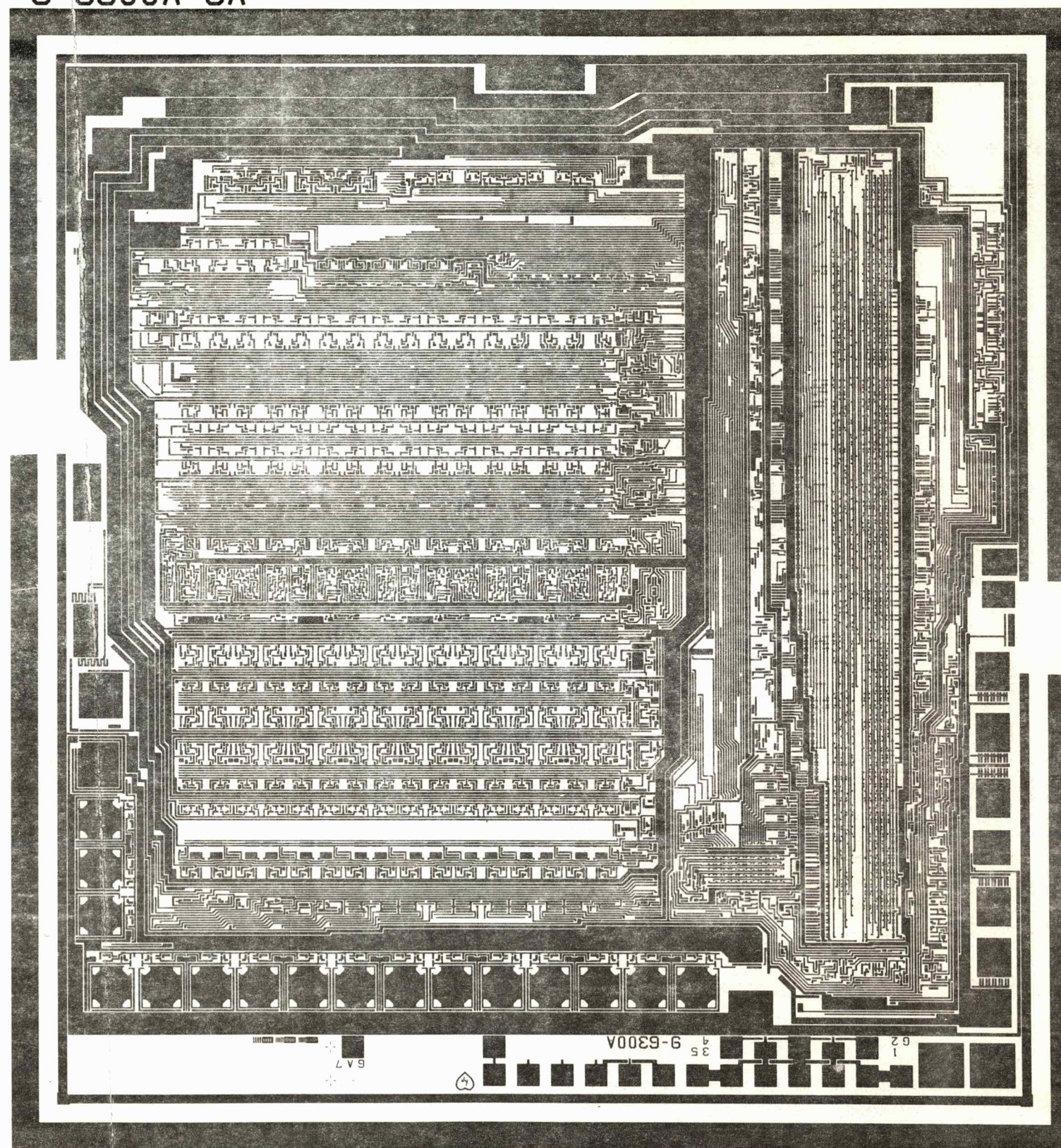
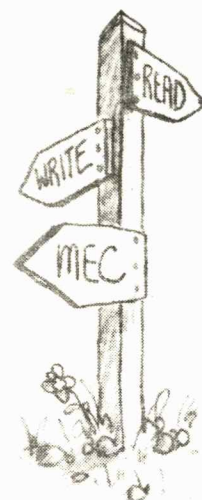
EMC ROM CONTENTS (CONT.)

EMC ROM CONTENTS (CONT.)

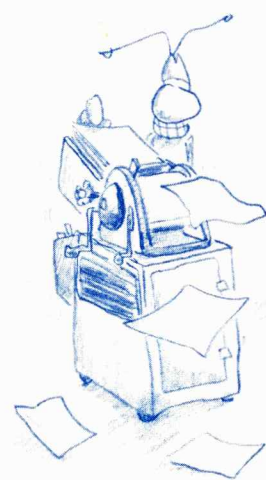
9-6300A-6A



MEMORY CYCLE



2-24-76



PRINTED
IN
USA

